

## 0062501A2 1 2

1 signal, includes logic to detect the second signal in the composite signal, an adaptive filter to  
filter the first signal, wherein the logic controls filter adaptation responsive to detecting the  
second signal in the composite signal, a difference operator to subtract the filtered first signal  
from the composite signal, and a delay between the logic and the difference operator.

5 In yet still another embodiment of the present invention, a method of synchronizing a data  
signal in a system having an input clock that is uncorrelated to an output clocks includes  
generating an error signal as a function of the difference in the input and output clock, and  
fractionally resampling the data signal as a function of the error signal.

## 10 DESCRIPTION OF THE DRAWINGS

These and other features, aspects, and advantages of the present invention will become  
better understood with regard to the following description, appended claims, and accompanying  
drawings where:

15 FIG. 1 is a schematic diagram of a hybrid fiber coaxial (HFC) network showing typical  
pathways for data transmission between the headend (which contains the cable modem  
termination system) and a plurality of homes (each of which contain a cable modem);

20 FIG. 2 is a simplified block diagram of a network gateway integrated into a cable modem  
system wherein the network gateway interfaces a plurality of packet based and circuit switched  
systems with a cable modem termination system (CMTS), defined as a line card, disposed at the  
headend;

FIG. 3 is a system block diagram of an exemplary network gateway in accordance with  
a preferred embodiment of the present invention;

FIG. 4 is a graphical depiction of the chaining mode of operation of the system direct  
memory access controller in accordance with a preferred embodiment of the present invention;

25 FIG. 5 is a system block diagram of a DOCSIS downstream demodulator in accordance  
with a preferred embodiment of the present invention;

FIG. 6 is a system block diagram of a DOCSIS upstream modulator in accordance with  
a preferred embodiment of the present invention;

30 FIG. 7 is a system block diagram of a DOCSIS media access controller (MAC) in  
accordance with a preferred embodiment of the present invention;

FIG. 8 is a system block diagram of an Ethernet transceiver for interfacing the network  
gateway with Ethernet devices in accordance with a preferred embodiment of the present  
invention;

35 FIG. 9 is a system block diagram of an Ethernet media access controller (MAC) in  
accordance with a preferred embodiment of the present invention;

FIG. 10 is a system block a Universal Serial Bus (USB) controller in accordance with a  
preferred embodiment of the present invention;

FIG. 11 is a block diagram of the analog front end for interfacing the analog processor

1 with an external subscriber line interface circuit (SLIC) in accordance with a preferred embodiment of the present invention;

FIG. 11A is a block diagram of an external interface between the analog front end and the subscriber line interface circuit (SLIC) in accordance with a preferred embodiment of the present invention;

FIG. 12 is a block diagram of the audio processor that interfaces the voice and data processor with external subscriber line circuits (SLICs) via the analog front end in accordance with a preferred embodiment of the present invention;

FIG. 13 is a block diagram of a ring generator that synthesizes a reference waveform that is utilized by external SLICs to ring telephony devices in accordance with a preferred embodiment of the present invention;

FIG. 14 is a system block diagram of a network gateway for interfacing between a hybrid fiber coaxial (HFC) network and a switched circuit network and a packet based network in accordance with a preferred embodiment of the present invention;

FIG. 14A is a block diagram of a timing recovery system for synchronizing the timing regeneration circuit clock of the network gateway to the CMTS clock in accordance with a preferred embodiment of the present invention;

FIG. 15 is a block diagram of a network gateway including a voice synchronizer for synchronizing voice data signals between telephony devices on the near and far ends of a HFC network in accordance with a preferred embodiment of the present invention;

FIG. 16 is a graphical depiction of fractional interpolation and decimation of a digitized analog voice signal in accordance with a preferred embodiment of the present invention;

FIG. 17 is a general block diagram of a voice synchronizer that generates an error signal which is used to polyphase re-sample the input voice signal so as to synchronize the near end signal with the far end signal in accordance with a preferred embodiment of the present invention;

FIG. 17A is a block diagram of clock divider circuits for generating various clocks for use within the network gateway in accordance with a preferred embodiment of the present invention;

FIG. 18 is a voice synchronizer for the upstream direction wherein the TRC clock drives a counter which is clocked by a high frequency ADC clock and the incoming voice signal is re-sampled in accordance with the ratio of the counter output divided by the ratio of the high frequency ADC clock and the TRC clock in accordance with a preferred embodiment of the present invention;

FIG. 18A is a block diagram of a single pole low pass filter used to smooth or average the differences between sampling rates in accordance with a preferred embodiment of the present invention;

FIG. 18B is a voice synchronizer for the downstream direction wherein a frame arrival clock drives a counter that is clocked by a high frequency DAC clock so that the incoming voice

1 signal is re-sampled in accordance with the ratio of the counter output divided by the ratio of the high frequency DAC clock and the frame arrival in accordance with a preferred embodiment of the present invention;

5 FIG. 19 is a block diagram of an echo canceller which utilizes energy estimates to detect near end speech in the presence of far end speech to control the adaptation of a adaptive filer in accordance with a preferred embodiment of the present invention;

10 FIG. 20 is a block diagram of a echo canceller which avoids delays in the detection of near end speech that are typically associated with conventional echo cancellers and utilizes the delay associated with a decimator and high pass filter to provide a look ahead capability so that filter adaptation may be disabled in advance of the arrival of near end speech in accordance with a preferred embodiment of the present invention;

15 FIG. 20A is a block diagram of an echo canceller that utilizes a delay buffer to provide a look ahead capability so that filter adaptation may be disabled in advance of the arrival of near end speech in accordance with a preferred embodiment of the present invention;

20 FIG. 21 is a block diagram of method for reducing the amplitude level of near end speech prior to double talk detection in accordance with a preferred embodiment of the present invention;

25 FIG. 22 is a block diagram of a signal processing system implemented with a programmable digital signal processor (DSP) software architecture for interfacing telephony devices with packet based networks in accordance with a preferred embodiment of the present invention;

30 FIG. 23 is a block diagram of the MTA architecture for interfacing the voice processor DSP with the MIPS core processor of the network gateway in accordance with a preferred embodiment of the present invention;

35 FIG. 24 is a block diagram of the architecture for gateway control protocol logic which interfaces with an external control element, called a call agent or call management server (CMS), to terminate and generate call signaling from and to the voice and data processor in compliance with the network-based call signaling (NCS) protocol specification, in accordance with a preferred embodiment of the present invention;

FIG. 25 is a block diagram of method for converting between real time protocol packets required by the PacketCable exchange specification and the xChange packet format utilized by the voice and data processor and vice versa in accordance with a preferred embodiment of the present invention;

FIG. 26 is a block diagram of software messaging interface between the host DSP and the voice and data processing software in accordance with a preferred embodiment of the present invention;

FIG. 27 is a block diagram of channel associated signaling service logic for exchanging commands and events between the host MTA call client and standard commercial analog



1 loop/ground start devices such as for example plain old telephone sets in accordance with a preferred embodiment of the present invention;

FIG. 28 is a block diagram of the software architecture operating on the DSP platform of FIG. 22 in accordance with a preferred embodiment of the present invention;

5 FIG. 29 is state machine diagram of the operational modes of a virtual device driver for packet based network applications in accordance with a preferred embodiment of the present invention;

FIG. 30 is a system block diagram of a signal processing system operating in a voice mode in accordance with a preferred embodiment of the present invention;

10 FIG. 31 is a block diagram of a method for canceling echo returns in accordance with a preferred embodiment of the present invention;

FIG. 32 is a block diagram of a method for normalizing the power level of a digital voice samples to ensure that the conversation is of an acceptable loudness in accordance with a preferred embodiment of the present invention;

15 FIG. 33 is a graphical depiction of a representative output of a peak tracker as a function of a typical input signal, demonstrating that the reference value that the peak tracker forwards to a gain calculator to adjust the power level of digital voice samples should preferably rise quickly if the signal amplitude increases, but decrement slowly if the signal amplitude decreases in accordance with a preferred embodiment of the present invention;

20 FIG. 34 is a graphical depiction of exemplary operating thresholds for adjusting the gain factor applied to digital voice samples to ensure that the conversation is of an acceptable loudness in accordance with a preferred embodiment of the present invention;

FIG. 35 is a block diagram of a method for estimating the spectral shape of the background noise of a voice transmission in accordance with a preferred embodiment of the present invention;

25 FIG. 36 is a block diagram of a method for generating comfort noise with an energy level and spectral shape that substantially matches the background noise of a voice transmission in accordance with a preferred embodiment of the present invention;

FIG. 37 is a block diagram of the voice decoder and the lost packet recovery engine in accordance with a preferred embodiment of the present invention;

30 FIG. 38A is a flow chart of the preferred lost frame recovery algorithm in accordance with a preferred embodiment of the present invention;

FIG. 38B is a flow chart of the voicing decision and pitch period calculation in accordance with a preferred embodiment of the present invention;

35 FIG. 38C is a flow chart demonstrating voicing synthesis performed when packets are lost and for the first decoded voice packet after a series of lost packets in accordance with a preferred embodiment of the present invention;

FIG. 39 is a block diagram of a method for detecting dual tone multi frequency tones in

1 accordance with a preferred embodiment of the present invention;

FIG. 39A is a block diagram of a method for reducing the instructions required to detect a valid dual tone and for pre-detecting a dual tone;

5 FIG. 40 is a block diagram of a signaling service for detecting precise tones in accordance with a preferred embodiment of the present invention;

FIG. 41 is a block diagram of a method for detecting the frequency of a precise tone in accordance with a preferred embodiment of the present invention;

10 FIG. 42 is state machine diagram of a power state machine which monitors the estimated power level within each of the precise tone frequency bands in accordance with a preferred embodiment of the present invention;

FIG. 43 is state machine diagram of a cadence state machine for monitoring the cadence (on/off times) of a precise tone in a voice signal in accordance with a preferred embodiment of the present invention;

15 FIG. 44 is a block diagram of a cadence processor for detecting precise tones in accordance with a preferred embodiment of the present invention;

FIG. 45 is a block diagram of resource manager interface with several VHD's and PXD's in accordance with a preferred embodiment of the present invention;

FIG. 46 is a system block diagram of a signal processing system operating in a real time fax relay mode in accordance with a preferred embodiment of the present invention;

20 FIG. 47 is a diagram of the message flow for a fax relay in non error control mode in accordance with a preferred embodiment of the present invention;

FIG. 48 is a flow diagram of a method for fax mode spoofing in accordance with a preferred embodiment of the present invention;

25 FIG. 49 is a system block diagram of a signal processing system operating in a modem relay mode in accordance with a preferred embodiment of the present invention;

FIG. 50 is a diagram of a relay sequence for V.32bis rate synchronization using rate re-negotiation in accordance with a preferred embodiment of the present invention;

30 FIG. 51 is a diagram of an alternate relay sequence for V.32bis rate synchronization whereby rate signals are used to align the connection rates at the two ends of the network without rate re-negotiation in accordance with a preferred embodiment of the present invention;

FIG. 52 is a system block diagram of a QAM data pump transmitter in accordance with a preferred embodiment of the present invention;

FIG. 53 is a system block diagram of a QAM data pump receiver in accordance with a preferred embodiment of the present invention;

35 FIG. 54 is a block diagram of a method for sampling a signal of symbols received in a data pump receiver in synchronism with the transmitter clock of a data pump transmitter in accordance with a preferred embodiment of the present invention;

FIG. 55 is a block diagram of a second order loop filter for reducing symbol clock jitter

1 in the timing recovery system of data pump receiver in accordance with a preferred embodiment of the present invention;

FIG. 56 is a block diagram of an alternate method for sampling a signal of symbols received in a data pump receiver in synchronism with the transmitter clock of a data pump transmitter in accordance with a preferred embodiment of the present invention;

FIG. 57 is a block diagram of an alternate method for sampling a signal of symbols received in a data pump receiver in synchronism with the transmitter clock of a data pump transmitter wherein a timing frequency offset compensator provides a fixed dc component to compensate for clock frequency offset present in the received signal in accordance with a preferred embodiment of the present invention;

FIG. 58 is a block diagram of a method for estimating the timing frequency offset required to sample a signal of symbols received in a data pump receiver in synchronism with the transmitter clock of a data pump transmitter in accordance with a preferred embodiment of the present invention;

FIG. 59 is a block diagram of a method for adjusting the gain of a data pump receiver (fax or modem) to compensate for variations in transmission channel conditions in accordance with a preferred embodiment of the present invention;; and

FIG. 60 is a block diagram of a method for detecting human speech in a telephony signal in accordance with a preferred embodiment of the present invention.

## 1 DETAILED DESCRIPTION OF THE INVENTION

5 In an exemplary embodiment of a cable modem system, a headend or cable modem termination system (CMTS) is located at a cable company facility and functions as a modem which services a large number of subscribers. Each subscriber has a cable modem (CM). Thus, the cable modem termination system must be capable of facilitating bidirectional communication with any desired one of the plurality of cable modems.

10 As used herein, the cable modem termination system (CMTS) is defined to include that portion of a headend which facilitates communication with a number of cable modems. A typical cable modem termination system includes a burst receiver, a continuous transmitter and a medium access control (MAC).

15 The cable modem termination system communicates with the cable modems via a hybrid fiber coaxial (HFC) network, wherein optical fiber provides communication to a number of fiber nodes and each fiber node typically serves approximately 500 to 2,000 subscribers, which communicate with the node via coaxial cable. The subscribers communicate with the fiber node via a common or shared coaxial cable.

20 The HFC network of a cable modem system utilizes a point-to-multipoint topology to facilitate communication between the cable modem termination system and the cable modems. Frequency domain multiple access (FDMA)/time division multiplexing (TDM) is used to facilitate communication from the cable modem termination system to each of the cable modems, i.e., in the downstream direction. Frequency domain multiple access (FDMA)/time domain multiple access (TDMA) is used to facilitate communication from each cable modem to the cable modem termination system, i.e., in the upstream direction.

25 The cable modem termination system includes a downstream modulator for facilitating the transmission of data communications to the cable modems and an upstream demodulator for facilitating the reception of data communications from the cable modems.

30 The downstream modulator of the cable modem termination system generally utilizes either 64 QAM or 256 QAM in a frequency band of 54 MHz to 860 MHz to provide a data rate of up to 56 Mbps. Since the upstream channel has a much lower data rate requirement, the upstream demodulator may use either QPSK or 16 QAM in a frequency range of 5 MHz to 42 MHz to provide a data rate of up to 10 Mbps.

35 Similarly, each cable modem includes an upstream modulator for facilitating the transmission of data to the cable modem termination system and a downstream demodulator for receiving data from the cable modem termination system. In the described exemplary embodiment, the upstream modulator of each cable modem typically uses either QPSK or 16 QAM within the 5 MHz to 42 MHz bandwidth of the upstream demodulator and the downstream demodulator of each cable modem may use either 64 QAM or 256 QAM in the 54 MHz to 860 MHz bandwidth of the downstream modulator (in North America). As those skilled in the art will appreciate, other modulation schemes may be used depending on the application.

1 In order to accomplish TDMA for upstream communication, time slots may be assigned  
to each cable modem within which cable modems having a message to send to the cable modem  
termination system. The assignment of such time slots is accomplished by providing a request  
contention area in the upstream data path within which the cable modems are permitted to  
5 contend in order to place a message which requests additional time in the upstream data path for  
the transmission of their message. The cable modem termination system responds to these  
requests by assigning time slots to the cable modems making such a request, so that as many of  
the cable modems as possible may transmit their messages to the cable modem termination  
system utilizing TDMA and so that the transmissions are performed without undesirable  
10 collisions.

Because of the use of TDMA, the cable modem termination system uses a burst receiver,  
rather than a continuous receiver, to receive packets of information from the cable modems via  
upstream communications. As those skilled in the art will appreciate, a continuous receiver may  
be utilized where generally continuous communications (as opposed to burst communications  
15 as in the present invention) are performed, so as to substantially maintain timing synchronization  
between the cable modem transmitter and the CMTS receiver, for proper reception of the  
communicated information. During continuous communications, timing recovery is a more  
straightforward process since signal acquisition generally only occurs at the initiation of such  
communications. Thus, acquisition is generally only performed in continuous receivers once per  
20 continuous transmission and each continuous transmission may be very long.

However, the burst communications inherent to TDMA systems entail periodic and  
frequent reacquisition of the signal. That is, during TDMA communications, the signal is  
reacquired for each separate burst transmission being received.

Referring now to FIG. 1, a hybrid fiber coaxial (HFC) network 10 facilitates the  
25 transmission of data between a headend 12, which includes at least one cable modem termination  
system, and a number of homes 14, each of which contains a cable modem. Such hybrid fiber  
coaxial networks are commonly utilized by cable providers to provide Internet access, cable  
television, pay-per-view and the like to subscribers.

Approximately 500 homes 14 are in electrical communication with each node 16, 34 of  
30 the hybrid fiber coaxial network 10, typically via coaxial cables 29, 30, 31. Amplifiers 15  
facilitate the electrical connection of the more distant homes 14 to the nodes 16, 34 by boosting  
the electrical signals so as to desirably enhance the signal-to-noise ratio of such communications  
and by then transmitting the electrical signals over coaxial cables 30, 31. Coaxial cable 29  
electrically interconnects the homes 14 with the coaxial cables 30, 31, which extend between  
35 amplifiers 15 and nodes 16, 34. Each node 16, 34 is electrically connected to a hub 22, 24,  
typically via an optical fiber 28, 32. The hubs 22, 24 are in communication with the headend 12,  
via optical fibers 20, 26. Each hub is typically capable of facilitating communication with  
approximately 20,000 homes 14.

1           The optical fibers 20, 26 extending intermediate the headend 12 and each hub 22, 24  
defines a fiber ring which is typically capable of facilitating communication between  
approximately 100,000 homes 14 and the headend 12. The headend 12 may include video  
servers, satellite receivers, video modulators, telephone switches and/or Internet routers 18, as  
5           well as the cable modem termination system. The headend 12 communicates via transmission  
line 13, which may be a T1 or T2 line, with the Internet, other headends and/or any other desired  
device(s) or network.

          In an exemplary embodiment of the present invention, a network gateway may facilitate  
on line and off line bi-directional communication between a number of near end data or  
10          telephony devices and far end data termination devices via a cable modem termination system.  
An exemplary topology is shown in FIG. 2 with a network gateway 11a providing an interface  
to HFC network 10a for a telephone 53a, fax machine 54 and modem 53c through a subscriber  
loop interface circuit (SLIC) 50. The network gateway 11a also provides an interface to the 10a  
HFC network for a telephone 53b, a fax machine 54b and a modem 55b through our audio  
15          processor 52. A local area network (LAN) 46 and a universal synchronous bus (USB) 44 can  
also be provided access to the HFC network 10a via the network gateway 11a. The HFC network  
is coupled to a CMTS line card 42 in the CMTS 12. The CMTS card 12 is coupled a packet  
based network router 40 to determine whether the communication will be transported via a far  
end HFC network 10b, a far end PSTN network 57 or the Internet 43. In the case of a far end  
20          PSTN network a PSTN gateway 58 provides an interface between a far end PSTN device 56 and  
a PSTN network 57 connected to the CMTS 12.

          In the case of a far end HFC network, a network gateway 11b provides an interface  
between the far end data termination devices 56B and the far end HFC network 10b connected  
to the CMTS 12. As those skilled in the art will appreciate, the far end data termination devices  
25          56 can include a variety of telephony and data devices including a telephone, fax machine, and  
modem via a SLIC or audio processor, as well as a LAN or USB.

          A packet based network router 40 facilitates communication between the near end data  
devices and off line far end data terminating devices 56 via a circuit switched networks such as  
the public switched telephone network (PSTN) 57 and a PSTN gateway 58. Data termination  
30          devices include by way of example, analog and digital phones, ethernet phones, Internet Protocol  
phones, fax machines, data modems, cable modems, interactive voice response systems, PBXs,  
key systems, and any other conventional telephony devices known in the art. One of skill in the  
art will appreciate that the described exemplary network gateway is not limited to a telephony  
interface to the HFC network, but rather may interface with a variety of communication media  
35          such as for example, asymmetric digital subscriber line (ADSL), DSL, T1 or other networks  
known in the art. Therefore, the described hybrid fiber coaxial network 10a is by way of example  
and not limitation.

          An exemplary embodiment of the network gateway is shown schematically in FIG. 3.

1 The described exemplary embodiment may provide a highly integrated solution implemented  
single chip that is compliant with the Data Over Cable Service Interface Specification (DOCSIS).  
DOCSIS was developed to ensure that cable modem equipment built by a variety of  
manufacturers is compatible, as is the case with traditional dial-up modems. The described  
5 exemplary embodiment can provide integrated functions for communicating with the CMTS.  
For example, a QPSK upstream modulator 102 transmits data to the far end data terminating  
device, a QAM downstream demodulator 100 receives data from the far end data terminating  
device via a CMTS, and a QPSK out of band downstream demodulator 106 receives out of band  
MPEG-2 encoded messages from the CMTS.

10 In addition, the described exemplary embodiment can support multiple inputs in  
accordance with a variety of protocols. For example, a universal serial bus transceiver 104  
provides transparent bi-directional IP traffic between devices operating on a USB such as for  
example a PC workstation, server printer or other similar devices and the far end data terminating  
device. Additionally, an I.E.E 802.3 compliant media independent interface (MII) 110 in  
15 conjunction with an Ethernet MAC 134 also provide bi-directional data exchange between  
devices such as, for example a number of PCs and or Ethernet phones and the far end data  
terminating device. A voice and data processor 160 is used for processing and exchanging voice,  
as well as fax and modem data between packet based networks and telephony devices.

20 The QAM downstream demodulator 100 may utilize either 64 QAM or 256 QAM in the  
54 to 860 MHz bandwidth to interface with the CMTS. The QAM downstream demodulator 100  
accepts an analog signal centered at the standard television IF frequencies, amplifies and digitizes  
the signal with an integrated programable gain amplifier and A/D converter. The digitized signal  
is demodulated with recovered clock and carrier timing. Matched filters and then adaptive filters  
remove multi-path propagation effects and narrowband co-channel interference. Soft decisions  
25 are then passed off to an ITU-T J.83 Annex A/B/C compatible decoder. The integrated decoder  
performs error correction and forwards the processed received data, in either parallel or serial  
MPEG-2 format to a DOCSIS Media Access Controller (MAC) 112.

30 The output of the downstream demodulator 100 is coupled to the DOCSIS MAC 112. The  
DOCSIS MAC 112 may include baseline privacy encryption and decryption as well as robust  
frame acquisition and multiplexing with MPEG2-TS compliant video and audio streams. The  
DOCSIS MAC 112 implements the downstream portions of the DOCSIS protocol. The DOCSIS  
MAC 112 extracts DOCSIS MAC frames from MPEG-2 frames, processes MAC headers, and  
filters and processes messages and data.

35 Downstream data packets and message packets may be then placed in system memory 114  
by a SDRAM interface 116 via the internal system bus 118. The SDRAM interface 116  
preferably interfaces to a number of off the shelf SDRAMs which are provided to support the  
high bandwidth requirements of the Ethernet MAC 112 and other peripherals. The SDRAM  
interface 116 may support multiple combinations of 8, 16 or 32 bit wide SDRAMs, allowing for

1 external data storage in the range of about 2 to 32 MBytes. The DOCSIS MAC 112 includes a number of direct memory access (DMA) channels for fast data access to and from the system memory 114 via the internal system bus 118.

5 The upstream modulator 102 provides an interface with the CMTS. The upstream modulator 102 may be configured to operate with numerous modulation schemes including QPSK and 16-QAM. The upstream modulator 102 supports bursts or continuous data, provides forward error correction (FEC) encoding and pre-equalization, filters and modulates the data stream and provides a direct 0-65 MHz analog output.

10 The DOCSIS MAC 112 can also implement the upstream portions of the DOCSIS protocol before transmission by the upstream modulator 102. The DOCSIS MAC 112 receives data from one of the DMA channels, requests bandwidth and frames the data for TDMA with other modems on the same upstream frequency.

15 The DOCSIS MAC interfaces with the MIPS core 128 via the ISB 118. An exemplary embodiment of the MIPS core 128 includes a high performance CPU operating at a speed of at least 80 MHz with 32-bit address and data paths. The MIPS core includes two way set associative instruction and data caches on the order of about 4kbytes each. The MIPS core 128 can provide standard EJTAG support with debug mode, run control, single step and software breakpoint instruction as well as additional optional EJTAG features.

20 The upstream modulator 102 and the downstream demodulator 100 are controlled by the MIPS core 128 via a serial interface which is compatible with a subset of the Motorola M-Bus and the Philips I<sup>2</sup>C bus. The interface consists of two signals, serial data (SDA) and serial clock (SCL), which may control a plurality of devices on a common bus. The addressing of the different devices may be accomplished in accordance with an established protocol on the two wire interface.

25 The described exemplary embodiment of the network gateway includes a full-speed universal serial bus (USB) transceiver 1104 and USB MAC 122 which is compliant with the USB 1.1 specification. The USB MAC 122 provide concurrent operation of control, bulk, isochronous and interrupt endpoints. The USB MAC 122 also can support standard USB commands as well as class/vendor specific commands. The USB MAC 122 include integrated RAM which allows flexible configuration of the device. Two way communication of information to a device operating on a USB can be provided, such as for example a PC on a USB 1.1 compliant twisted pair. The USB MAC 122 can be arranged for hardware fragmentation of higher layer packets from USB packets with automatic generation and detection of zero length USB packets. The USB MAC 122 may include DMA channels which are used to communicate received data to the system memory 114 via the internal system bus 118. Data stored in system memory 114 may then be processed and communicated to the cable modem termination system(not shown) via the DOCSIS MAC 112 and the upstream modulator 102. Similarly data received from the cable modem termination system and processed by the downstream



1 demodulator 100 and stored in system memory as higher layer packets can be retrieved by the  
USB MAC122 via the ISB 118 and assembled into USB packets with automatic generation of  
zero length USB packets. USB packets may then be communicated to the external device  
operating on the USB via the USB transceiver 104.

5 A media independent interface (MII) 110 can provide bi-directional communication with  
devices such as for example a personal computer (PC) operating on an Ethernet. The media  
independent interface 110 can forward data to and receive information from the Ethernet MAC  
134. The Ethernet MAC 134 can also perform all the physical layer interface (PHY) functions  
for 100BASE-TX full duplex or half-duplex Ethernet as well as 10BASE-T full or half duplex.

10 The Ethernet MAC 134 can also decode the received data in accordance with a variety of  
standards such as for example 4B5b, MLT3, and Manchester decoding. The Ethernet MAC can  
perform clock and data recovery, stream cipher de-scrambling, and digital adaptive equalization.  
The Ethernet MAC 134 may include DMA channels which are used for fast data communication  
of processed data to the system memory 114 via the internal system bus 118. Processed data  
15 stored in system memory 114 may then be communicated to the cable modem termination  
system(not shown) via the upstream modulator 102. Similarly, data received from the cable  
modem termination system is processed by the downstream demodulator 100 and stored in  
system memory as higher layer packets which can then be retrieved by the Ethernet MAC 1134  
via the ISB 118 and encoded into Ethernet packets for communication to the external device  
20 operating on the Ethernet via the MII 110. The Ethernet MAC 134 may also perform additional  
management functions such as link integrity monitoring, etc.

In addition to the SDRAM interface 116, the described exemplary embodiment of the  
gateway includes a 16-bit external bus interface (EBI) 140 that supports connection to flash  
memories 142, external SRAM 144 or EPROMS 144. Additionally, the EBI 140 may be used  
25 to interface the described exemplary network gateway with additional external peripherals. The  
EBI 140 can provide a 24 bit address bus and a 16-bit bi-directional data bus. Separate read and  
write strobes can be provided along with multiple firmware configurable chip select signals.  
Each chip select can be fully programmable, supporting block sizes between about 4 K-bytes and  
8 M-bytes, extended clock cycle access control and 8 or 16-bit selection of peripheral data bus  
30 width. In the described embodiment, the EBI 140 can support both synchronous and  
asynchronous transfers. Pseudonymous transfers may be supported through the use of read/write  
strobes to indicate the start and duration of a transfer. The EBI 140 can include DMA access  
capability to or from the SDRAM interface 116. The DMA operation may take one or more  
forms. For example, in EBI mode, an EBI bridge can act as the DMA controller, and perform  
35 all pointer and buffer management tasks during DMA operations. In an external mode, an  
external device can act as the DMA controller and the EBI 140 can serve as a simple bridge. In  
DMA mode the MIPS core128 can be responsible for DMA setup.

The network gateway may be vulnerable to network breaches due to peripheral devices

1 such as PC employing windows or network Macintosh computers. These operating systems  
include "file sharing" and "printer sharing" which allow two or more networked computers in a  
home or office to share files and printers. Therefore the exemplary embodiment of the gateway  
includes IP security module 1148 which interfaces with ISB 118. The MIPS core128 can set-up  
5 and maintain all security associations. The MIPS core128 can also filter all IP traffic and route  
any messages requiring security processing to the security module via the ISB 118. The security  
module 150 may support single DES (CBC and ECB modes) triple DES (CBC and ECB modes)  
MD-5 and SHA authentication in hardware to provide support for virtual private networks.

The security module 148 can implement the basic building blocks of the developing IP  
10 Security Standard (IPsec). The security module 148 may also be used to implement any other  
security scheme that uses the same basic blocks as IPsec, which uses two protocols to provide  
traffic security. A first protocol, IP Encapsulating Security Payload (ESP), provides private data  
privacy with encryption and limited traffic flow confidentiality. ESP may also provide  
connection less integrity, data source authentication and an anti-replay service. A second format,  
15 IP Authentication Header (AH), provides connection less integrity, data source authentication and  
an optical anti-replay service. Both protocols may be used to provide access based on the  
distribution of cryptographic keys and the management of traffic flows. The protocols may be  
used alone or in combination to satisfy the security requirements of a particular system. In  
addition, the security module 148 can support multiple modes of operation depending on a  
20 security association to the traffic carried by a simplex connection. For example, transport mode  
security association between two hosts, primarily protects protocols above the IP layer while  
tunnel mode security association provides security and control to a tunnel of IP packets.

The exemplary security module 148 addresses possible differences in packet format  
between IPsec and future security applications with a generalized scheme to determine where the  
25 authentication / encryption algorithms are applied with a data packet. The authentication /  
encryption algorithms consider each packet to consists of three parts, a header, body and trailer.  
The appropriate algorithm can be applied, using any specified parameters to the body section  
only.

In an encryption mode, the security module 148 can add and initialize any necessary  
30 headers, determine necessary parameters, generate the associated control message and add the  
control and data message. In the authentication mode, the control fields of the received data  
packets are parsed, the parameters are determined via a security association lookup table, control  
message is created and the control and data messages are enqueued.

The exemplary embodiment of the network gateway includes a DMA controller 150  
35 having a number of channels that enable direct access over the ISB 118 between peripherals and  
the system memory 114. With the exception of the security module 148, packets received by the  
network gateway 98 cause DMA transfers from a peripheral to memory, which is referred to as  
a receive operation. A DMA transfer from memory to a peripheral is referred to as a transmit

1 operation. Programmable features in each channel can allow DMA controller 150 to manage maximum ISB burst lengths for each channel, enable interrupts, halt operation in each channel, and save power when certain modules are not operational. The maximum ISB burst length may be programmed independently for each channel preferably up to 64 32 bit words. Each channel  
5 can include maskable interrupts connected to the MIPS core128 which indicate buffer complete, packet complete and or invalid descriptor detected. Busy DMA channels may be stalled or completely disabled by the MIPS core128. Source clocks (not shown) for each channel are can be connected to the channels based on the internal peripheral they service. For power reduction, these clocks may be turned off and on coincident with the respective peripheral's clock.

10 The DMA controller 150 can be operable in both non-chaining and chaining mode. In the non-chaining mode the DMA channel refers to its internal registers for the pertinent information related to a scheduled DMA burst transfer. The DMA controller can set-up the buffer start address, byte count, and status word registers before initiating the DMA channel for each allocated buffer. In the transmit direction, the DMA channels can send the specified number  
15 of bytes (preferably up to 4095) from the specified byte address. In the receive direction, the DMA channels can insert data into a specified memory location until a buffer has been completely filled or the end of a packet is detected.

In the chaining mode, the system memory can be partitioned as shown in FIG. 4 preferably using descriptor rings containing pointers to memory buffers as well as status  
20 information for each memory buffer. The MIPS core128 can write the descriptor pointers while the DMA controller 150 follows by inserting/taking data into/from the location designated by the descriptor. Upon completion of the transfer of a buffer, the DMA controller 150 effectively clears the descriptor by updating the status to indicate that the data has been inserted/taken. Specific information may be added to the descriptor to indicate the length of data in the block,  
25 specifying whether the data is the first or last block of a packet, etc.

In the downstream direction, the MIPS core 128 can fill or recognize a data block for a particular DMA channel, then write the next unused descriptor in the ring indicating that the block is filled and where the downstream data exists in memory. The DMA controller 150 can follow the DSP write to the descriptor ring, sending out data and clearing the descriptor when the  
30 transfer is complete. When the DMA controller 150 reads a descriptor that does not contain valid data, it can go idle until initiated by the DSP core.

In the upstream direction, the MIPS core128, can allocates memory space for incoming data, then write the descriptor with the start address for that buffer. The DMA controller 150 read the base address and insert data until either the buffer is full or an end of packet has been  
35 detected. The DMA controller 150 can update the descriptor, communicating to the MIPS core128 that the block is full, indicating the length of the data on the block, and/or asserted first and or last buffer flags.

The described exemplary network gateway can include a voice processor 160 for

1 processing and transporting voice over packet based networks such as PCs running network on  
a USB (Universal Serial Bus) or an asynchronous serial interface, Local Area Networks (LAN)  
such as Ethernet, Wide Area Networks (WAN) such as Internet Protocol (IP), Frame Relay (FR),  
Asynchronous Transfer Mode (ATM), Public Digital Cellular Network such as TDMA (IS-13x),  
5 CDMA (IS-9x) or GSM for terrestrial wireless applications, or any other packet based system.  
The described embodiment of the voice processor 160 also supports the exchange of voice, as  
well as fax and modem, between a traditional circuit switched network or any number of  
telephony devices and the CMTS (not shown). The voice processor may be implemented with  
a variety of technologies including, by way of example, embedded communications software that  
10 enables transmission of voice over packet based networks.

The embedded communications software is preferably run on a programmable digital  
signal processor (DSP). In an exemplary embodiment the voice processor 160 utilizes a ZSP core  
from LSI Logic Core ware library for mid to high end telecommunications applications. The DSP  
core 160 can include at least about 80k words internal instruction RAM 162 and at least about  
15 48k words internal data RAM 164. The DSP core 160 can interface with the internal system bus  
118 via a DSP/ISB interface 166 and the peripheral bus 132 via the DSP/PB interface 168.

The DSP core 160 can provide a JTAG Emulator interface as well as internal training  
recovery clock (TRC) sync interface. The voice processor 160 can include a grant synchronizer  
that insures timely delivery of voice signals to the MIPS core 128 for upstream transmission. In  
20 addition, a PCM interface 170 can provide the voice processor 160 with an interface to an  
internal audio processor 170 as well as an external audio processing circuits to support constant  
bit rate (CBR) services such as telephony. The PCM interface can provide multiple PCM  
channel controllers to support multiple voice channels. In the described exemplary embodiment  
of the gateway, there are four sets of transmit and receive FIFO registers, one for each of the four  
25 PCM controllers. However, the actual number of channels that may be processed may vary and  
is limited only by the performance of the DSP. The internal system bus 118 is used to transfer  
data, control and status messages between the voice processor 160 and the MIPS core 128.  
FIFO registers are preferably used in each direction to store data packets.

The described exemplary embodiment of the gateway includes an internal audio processor  
30 170 with an analog front end 172 which interface the voice processor 169 with external  
subscriber line interface circuits (SLICs) for bi-directional exchange of voice signals. The audio  
processor 170 may include programable elements that implement filters and other interface  
components for a plurality of voice channels. In the transmit mode the analog front end 172  
accepts an analog voice signal and digitizes the signal and forwards the digitized signal to the  
35 audio processor 170.

In the described exemplary embodiment, the audio processor 170 may include A-law /  
 $\mu$ -law (G.711 compatible) encoder and decoder decimate the digitized signal and condition the  
decimated signal to remove far end echos.

As the name implies, echos in telephone systems is the return of the talker's voice resulting from the operation of the hybrid with its two-four wire conversion. If there is low end-to-end delay, echo from the far end is equivalent to side-tone (echo from the near-end), and therefore, not a problem. Side-tone gives users feedback as to how loud they are talking, and indeed, without side-tone, users tend to talk too loud. However, far end echo delays of more than about 10 to 30 msec significantly degrade the voice quality and are a major annoyance to the user. The audio processor can apply a fixed gain / attenuation to the conditioned signal and forwards the gain adjusted signal to the voice processor 160 via the PCM interface. In the receive mode the audio processor accepts a voice signal from the PCM interface and preferably applies a fixed gain/attenuation to the received signal. The gain adjusted signal is then interpolated from 8kHz to 96 kHz before being D/A converted for communication via a SLIC interface to a telephony device.

Each audio channel can be routed to a PCM port to allow for system level PCM testing. The PCM system tests, by way of example may require compliance with ITU G.711 for A-law and  $\mu$ -law encoding / decoding.

The described exemplary embodiment of the network gateway include integrated peripherals including independent periodic interval timers 180, a dual universal asynchronous receiver-transmitter (UART) 182 that handles asynchronous serial communication, a number of internal interrupt sources 184, and a GPIO module 186 that provides multiple individually configurable input/output ports. In addition, multiple GPIO ports can be provided to drive various light emitting diodes (LEDs) and to control a number of external SLICs. A peripheral bus bridge 186 can be used to interface the low speed peripheral to the internal system bus 118.

#### A. DOCSIS Cable Modem

##### 1. Downstream Demodulator

The DOCSIS downstream demodulator 100 can support 64/256 QAM. Referring to FIG. 5 the downstream demodulator 100 accepts an analog IF input signal 198, amplifies and digitizes the input signal with an integrated programable gain amplifier (PGA) 200, and an bit A/D converter 202. An on chip gain recovery loop 204 circuit controls the PGA 200 to provide an on chip automatic gain control (AGC) function. The timing recovery also includes an on chip voltage controlled oscillator (not shown) which can be locked to an off chip crystal, controls the sub-sampling of the A/D converter 202. The stability of the crystal reference allows for accurate sub-sampling of the IF signal to produce a digital data stream centered on a lower IF center frequency.

A digital demodulator 208 demodulates the digitized output 202(a) of the A/D converter 202, with recovered clock and carrier timing. The digital demodulator 208 includes digital mixers 210, 212 which mix a complex data stream generated by a direct digital frequency synthesizer (DDFS) 211 under the control of the timing recovery loop with the digitized signal 202(a). Matched interpolation filters 214, 216 convert mixer outputs 214(a), 216(a) to in-phase

1 (I) and quadrature-phase(Q) baseband signals correctly sampled in both frequency and phase. Dual square root Nyquist filters 218, 220 which may accommodate 11-18% roll-off factors, filter the I & Q baseband signals to reduce inter-symbol interference. In addition, notch filters 222, 224 may be used to substantially reduce narrowband co-channel interference caused by  
5 intermodulation products from analog CATV channels. Notch filters 222, 224 preferably place notches in the frequency spectrum at the frequencies of these subcarriers.

The downstream demodulator 102 preferably includes a configurable multi-tap decision directed equalizer 226. In the described exemplary embodiment, a 40 tap equalizer is used to remove intersymbol interference generated by worst case coaxial cable channels with multipath  
10 spreads of up to 4.5µsec at 5.26 Mbaud. Blind convergence algorithms facilitate equalizer acquisition.

In addition to adaptive equalization, the configurable multi-tap decision directed equalizer 226 performs phase recovery on the equalized constellation points using a quadrature synthesizer and complex mixer under the control of the carrier recovery loop to track out residual carrier  
15 offsets and instantaneous phase offsets such as those caused by tuner microphonics. The output of the adaptive equalizer phase recovery block is forwarded to a forward error correction (FEC) decoder 228. The FEC decoder can support numerous decoders including ITU-T J.83 Annex A/B/C compatible decoders.

The Annex A/C decoder consists of four major functions, frame synchronization, convolution de-interleaving, Reed-Solomon error correction and de-randomization. Hard  
20 decisions are preferably input into the frame synchronization block which locks onto the inverted sync bit pattern. The MIPS core 128 sets the characteristics of the frame synchronization acquisition and retention via a CPU interface 230. Once synchronized, data interleaving is removed by the convolution de-interleaver 232. The de-interleaver can be based on the Ramsey  
25 III approach, and can be programmable to provide depths from  $I=1-204$  with  $J=204/I$ . An on chip RAM 234 can provide for  $I=1-12$ . After de-interleaving the data symbols are processed by a Reed-Solomon decoder, which can correct up to eight symbol errors per RS block. The decoded symbols are then de-randomized, which substantially undoes the randomization inserted at the modulator. The de-randomized output consists of MPEG-2 serial or parallel data, packet sync and  
30 a data clock.

The Annex B decoder includes five layers, trellis decoding, de-randomization, convolution de-interleaving, Reed-Solomon decoding and checksum decoding. The Annex B concatenated coding scheme along with interleaving provides good coding gain to combat gaussian noise while still protecting against burst errors. Soft decisions from the adaptive  
35 equalizer 226 are input to the trellis decoder which estimates the maximum likelihood of a sequence. The output sequences are forwarded to a frame synchronization and de-randomization block similar to those described for the Annex A/C decoders. A Reed-Solomon decoder preferably corrects up to three symbol errors per RS block. The checksum decoder accurately

1 identifies block containing uncorrectable errors. The downstream demodulator outputs MPEG-2 serial or parallel data, packet sync and a data clock to the DOCSIS MAC.

5 The downstream demodulator 100 also includes two AGC loops which provide control for both RF and IF variable gain amplifiers (VGAs). The gain control allocated to each loop may be established via a CPU interface 230. The RF gain control loop 236 may be closed at the tuner AGC 236 while the IF loop 204 may be completed with either an off chip VGA (not shown) or preferably with the internal PGA 200. The power of the internal IF signal is estimated and compared to a programmable threshold. If the estimated power exceeds the threshold, the appropriate AGC integrator is incremented by a programmable value. If the estimated power is less than the threshold, the appropriate AGC integrator is decremented by a comparable amount. The timing recovery loop 206 may include a timing error discriminant, a loop filter, and a digital timing recovery block which controls the digital re-sampler. The carrier frequency/phase recovery and tracking loops are all digital loops which simultaneously offer a wide acquisition range and a large phase noise tracking ability. The loops may use decision directed techniques to estimate the angle and direction for phase/frequency compensation. The loops can be filtered by integral-plus-proportional filters, in which the integrator and linear coefficients of the filter are programmable to provide the means of setting the loop bandwidths. The upper bits of the loop filter can control the direct frequency synthesizer 210, providing both accurate frequency generation and fine phase resolution.

20 The downstream demodulator 100 uses an on chip VCO (not shown) referenced to a single off chip crystal which can provide all required chip clocks. In addition, a spare D/A demodulator provides a 1-bit pulse-width modulated signal which may be used with an off chip RC filter. In addition, the downstream modulator may provide tuner control ports 238 which may be used to program two serially controlled tuner phase locked loop (PLL) frequency synthesizers.

## 25 2. Upstream Modulator

Referring to FIG. 6, the upstream modulator 102 can support QPSK and 16-QAM processing of burst or continuous data signals 102(a) received from the DOCSIS MAC. Burst encoding logic 240 includes FIFO registers and a FEC encoder, preamble preend and symbol mapper. The burst FIFO register decouples the input data rate from the transmission data rate and allows data to be input while a burst is being actively transmitted. The FEC encoder processes data stored in the FIFO. The FEC encoder may be a Reed-Solomon encoder with data randomization. The parallel to serial conversion of bytes entering the randomizer and serial to parallel conversion of bits leaving the randomizer may be programmed to be most significant bit (MSB) or least significant bit (LSB) first. The encoder may be programmed to correct from zero to ten symbols errors per RS block. The FEC encoder may also be configured to integrate the randomizer before or after the RS encoder. A programmable preamble of up to 1024 bits may then be added to the data burst and the completed data burst can be then mapped into 90 degree DQPSK, QPSK or 16-QAM.

1 The output of the burst encoding logic 240 is coupled to a pre-equalizer 244 which may  
be selectively enabled to pre-distort the transmitted waveform to offset the effects of inter-symbol  
interference (ISI). The data burst is then shaped by square root Nyquist filters 246, 248 which  
5 may have a selectable excess bandwidth factor of 25% or 50 %. The maximum passband ripple  
of these filters is preferably less than about 0.05 dB and the minimum stopband attenuation is  
preferably greater than about 60 dB. The shaped signals are forwarded to interpolation filter  
banks 250, 252 which interpolate the signal to the sample rate. The outputs of these filters are  
then mixed or modulated onto quadrature carriers generated by a digitally tunable frequency  
10 synthesizer 258 by mixers 254, 256. The I and Q components are then combined by summer  
260. The summer 260 outputs a digital waveform carrying the data burst whose spectrum is  
preferably centered on the desired RF frequency to a D/A converter 262. The D/A converter  
converts the digital, shaped output burst to an analog waveform. The D/A converter 262 may  
have a sample rate of up to about 200 MHz. A programmable gain attenuator 264 can be used  
to provide up to about 25 dB attenuation in steps on the order of about 0.4 dB.

### 15 3. DOCSIS MAC

The DOCSIS media access controller (MAC), includes baseline privacy encryption and  
decryption, transmission convergence support, a TDM/TDMA framer, and a scatter/gather DMA  
interface. The transmission convergence sub-layer supports robust frame acquisition and  
multiplexing with MPEG-TS compliant video and audio streams. The TDM/TDMA preferably  
20 handles time synchronization with the cable modem termination system, upstream MAP  
decoding, bandwidth request generation and contention resolution. The DOCSIS MAC may be  
divided into four major functions, downstream functions, upstream functions, DMA interface and  
miscellaneous control. The downstream functions of the DOCSIS MAC include receiving  
MPEG frames 100(b) from the downstream demodulator, extracting the DOCSIS MAC frames  
25 from the MPEG frames, processing the MAC header, filtering messages and data, processing  
MAP and SYNC messages, decrypting data packets if necessary and providing cyclic redundancy  
checks (CRCs) on the MAC payloads.

Referring to FIG. 7, a downstream processor 280 can include a physical layer (PHY)  
interface which provides the interface to the downstream demodulator (not shown). The PHY  
30 receives the incoming MPEG stream, filters on the predefined PID, and uses the offset value  
contained in the MPEG frame to find the MAC frames. The extracted MAC frames pass to a  
MAC header processing block in a message processor 282 and through a rate conversion FIFO  
to a MAC header processing block in a downstream data encryption security (DES) processor  
284 which provides baseline security by decrypting QAM downstream traffic. The MAC header  
35 processing blocks (not shown) examine the MAC header for type, wherein the MAC header  
processor in the message processor 282 processes only MAC messages while the MAC header  
processor in the downstream DES 284 processes packets that are not MAC messages.

The incoming MAC header is parsed for the presence of an extended header field. If



1 the extended header field is present, the MAC header processor block parses the extended header  
type-length-value (TLV) fields, and searches for baseline privacy header. Once the baseline  
privacy header has been located, the MAC header processor forwards the associated value data  
to the downstream DES 284 for further parsing and interpretation. With the exception of the  
5 baseline privacy extended header, all other header types are preferably ignored by the MAC  
header processor. Both MAC header processing blocks determine the length of the extended  
header, calculate the header check sequence (HCS) over the MAC header and compare the  
calculated HCS with that contained in the MAC header. In the described exemplary  
embodiment, if the calculated HCS values do not match the HCS values contained in the MAC  
10 header, the MAC processor preferably discards the packets and signals the PHY interface  
requesting re-synchronization. Those packets where the calculated HCS values match the values  
contained in the MAC header, the MAC header processor preferably generates control signals  
to flag the type of packet, the beginning of the extended header, the beginning of the protocol  
data unit (PDU) and the length of the PDU. The MAC header processor routes all PDUs  
15 matching the network gateway extended header type to the downstream DES 284 for decryption.

The message processor 282 calculates the CRC over the message payload in accordance  
with the control signals generated by the MAC header processor and supplies data to the DMA  
controller 290. There are a number of fault conditions on the DMA interface that require specific  
20 action. If an HCS fail is generated by the MAC header processor the DMA is prematurely  
terminated. If the CRC is correct the message processor 282 preferably examines the type field  
of the message. The message processor 282 extracts the time stamp from SYNC messages and  
passes these to the timing regeneration circuit 286. The timing regeneration circuit 286 provides  
the timing recovery using the time stamp values from the sync messages. In addition, the  
25 message processor 282 forwards messages that match the upstream channel ID and UCD change  
count to a MAP processor 288. The remaining messages with valid CRC are passed to a  
downstream message DMA 290 through a rate adjustment FIFO. If a FIFO full state is  
encountered, the DMA discards the current packet, flushes the FIFO and waits until the next  
packet arrives.

30 The downstream DES 284 receives data packets and control signals from the header  
processor in the downstream processor 280. The downstream DES 284 determines what type of  
filtering should be applied to the packet based on information contained in the MAC header. If  
a packet supports baseline privacy the downstream DES filters the packet based on the silence  
identifier (SID). The DES 284 preferably decrypts packets using the key corresponding to the  
35 SID and even/odd key bit in the baseline privacy header. The DOCSIS MAC does not perform  
decryption on packets that do not contain a baseline privacy header. The DES 284 preferably  
filters packets based upon the destination address and forwards the filtered packets to a CRC  
processor (not shown) which calculates a CRC-32 over the entire PDU payload. If the calculated

1 CRC does not match the value stored in the packet. a CRC error flag is set and the packet is marked as erred in the downstream DMA buffer.

5 The downstream DMA 290 is used to transfer data to system memory (not shown). The downstream DMA 290 may have two channels, one of which is used to transfer data into system memory, the other is used to transfer message packets into memory. The two channels can be substantially identical in function. The downstream DMA can use structures in the memory to control transfer of information. The structures can be established and maintained by firmware. Data can be stored in regions of memory called particles. One structure contains a ring of buffer descriptors with each buffer descriptor describing particles in the shared memory. 10 The DMA can store received downstream data in the particles and update the status and length in the buffer descriptor. The other structure is an additional ring of descriptors used to refer to single packets. Where a packet may be contained in any number of particles described by a like number of buffer descriptors, there is only one packet descriptor associated with the packet.

15 The upstream DMA 292 can include 16 upstream channels which read upstream packet headers and protocol data units (PDUs) from system memory. The upstream DMA 292 can preferably insert the HCS, CRC and piggyback fields when transferring packets to an upstream DES 294. The upstream DES 294 examines the extended header to determine if encryption is enabled. If encryption is not enabled, the upstream DES 294 forwards the packet to the upstream processor 296, otherwise if encryption is enabled the upstream DES 292 preferably uses the SID and even odd key bit in the extended header to perform a key lookup. The upstream DES 292 then encrypts the packet and forwards the packet to the upstream processor 294. The upstream processor 294 extracts MAC information elements (IEs) from the MAP FIFO and compares the elements to a next upstream minislot count. If the counts match, the upstream processor 294, evaluates the type of slot described by the MAP information element. If the network gateway needs to respond to the information element, the upstream processor 294 preferably loads the appropriate physical layer parameters to the upstream modulator 102 (see FIG. 3) and forwards the appropriate message or data packet. This operation depends on the slot type and status of the network gateway. The upstream processor 296 preferably responds to initial maintenance slots and will ignore all station maintenance slots and requests until the MIPS core 128 (see FIG. 3) signals that the network gateway has been initially ranged and assigned a SID (temporary or permanent). The upstream processor 296 responds to initial maintenance slots by extracting the ranging request message from internal memory that is programmed by the MIPS core. Once the network gateway has been initially ranged, the upstream processor 296 responds to station messages, requests, request/data regions, and grants and not to initial maintenance messages. 25 30 35 \

Once ranged, the upstream processor 296, monitors the upstream DMA queues. In the described exemplary embodiment there are 16 independent upstream queues maintained by the upstream DMA 292. When a packet is ready for transmission, the upstream processor randomly

1 selects one of the available requests slots. The upstream processor 296 then waits to receive a  
grant, an acknowledgment (grant with length zero) or an ACK time past the time it sent its  
request upstream. If the upstream processor 296 does not receive an acknowledgment or grant  
before the ACK time has past, the upstream processor 296 preferably uses truncated binary  
5 exponential back-off to select another request slot to resend its request. Once granted, the  
upstream processor 296, signals the upstream DMA 292 to begin extracting the packet from  
memory and forwarding the packet to the upstream DES for processing as described above. The  
upstream processor 296 then forwards the packet to the upstream modulator (not shown) at the  
appropriate transmission time.

#### 10 4. Integrated Ethernet MAC

Referring to FIG. 3, the integrated Ethernet MAC performs all the physical layer interface  
functions for 100BASE-TX full duplex or half duplex Ethernet on CAT 5 twisted pair and  
10BASE-T full or half duplex Ethernet on CAT 3, 4 or 5 twisted pair. The Ethernet MAC 122  
may be coupled directly to the Ethernet network via the media independent interface 110. In  
15 100BASE-TX mode, the Ethernet MAC 122 preferably monitors the twisted pair for transitions  
in the energy level of the received signal. Signals can be qualified using squelch detect circuits.  
The Ethernet MAC enters and remains in a "link fail" state when valid signals are not detected  
on the receive twisted pair. In the link fail state, the Ethernet MAC 122 preferably transmits only  
idle codes. When the Ethernet MAC 122 detects a valid signal for a predetermined period, the  
20 Ethernet MAC 122 enters a link pass state and the appropriate transmit and receive functions are  
enabled.

The Ethernet MAC 122 preferably provides the ability to negotiate its mode of operation  
over the twisted pair link using the auto negotiation mechanisms defined in the IEEE 802.3u  
specifications, the contents of which are incorporated herein by reference as if set forth in full.  
25 Auto-negotiation should be selectively enabled by the Ethernet MAC. When enabled, the  
Ethernet MAC 122 preferably chooses a mode of operation by advertising its abilities and  
comparing those abilities to those received from its link partner.

FIG. 8 shows the physical interface portion of the Ethernet MAC 122. The Ethernet MAC  
122 may perform 4B5B, MLT3, and Manchester encoding and decoding. For 100BASE-TX  
30 mode the Ethernet MAC enables a 4B5B encoder 316. The transmit packet is encapsulated by  
replacing the first two nibbles with a start of stream delimiter and appending an end of stream  
delimiter to the end of the packet. The transmitter will repeatedly send the idle code group  
between packets. When the MII transmit enable is asserted, data is inserted into the transmit data  
stream. The encoded data stream is scrambled by a stream cipher scrambler 318 to reduce  
35 radiated emissions on the twisted pair, serialized by serializer 320 and encoded into MLT3 signal  
levels. A multimode transmit digital to analog converter (DAC) 322 can be used to drive the  
MLT3 data onto the twisted pair cable. The multi-mode DAC can transmit MLT3-coded  
symbols in 100Base-TX mode and Manchester coded symbols in 10BASE-TX mode. The DAC

1 can perform programmable edge rate control in transmit mode, which decreases unwanted high frequency signal components. High frequency pre-emphasis is preferably performed in 10BASE-TX mode.

5 The Ethernet MAC 122 can receive a continuous data stream on twisted pair. A 100BASE-TX data stream is not always DC balanced, because the receive signal must pass through a transformer, and therefore, the DC offset of the differential receive input may wander. Baseline wander may reduce the noise immunity of the receiver. Wander Correction Logic 300 can be used to compensate for baseline wander by removing the DC offset from the input signal, thereby reducing the receive symbol error rate. Each receive channel of the Ethernet MAC 10 can include an analog to digital converter (ADC) 302. Ethernet MAC, exemplary the ADC 302 is a 6-bit, 125 MHz ADC, which samples the incoming analog data on the receive channel and produces a 6-bit output. The ADC output is forwarded to a digital adaptive equalizer 304 which removes interzonal interference created by the transmission channel media. The Ethernet MAC 122 combines feed forward equalization and decision feedback equalization to achieve an 15 optimal signal to noise ratio. The digital design increases tolerances to on chip noise and the filter coefficients preferably self adapt to any quality of cable or cable length. The adaptive equalizer can be bypassed in 10BASE-TX mode which includes transmit pre-equalization capability.

20 The Ethernet MAC 122 can be include a digital clock recovery / generator system 306 which creates all internal transmit and receive clocks. The transmit clocks cab be synched to a 50 MHz clock input while the receive clocks are locked to the incoming data streams. Clock recovery circuits optimized to MLT3, and Manchester encoding schemes can be included for use with each of three different operating modes. The input data streams are can be sampled by the recovered clock from each port and fed synchronously to the respective digital adaptive equalizer.

25 Following baseline wander correction, adaptive equalization, and clock recovery a data converter 308 converts the received data stream from MLT3 to NRZ data. The NRZ data is descrambled by a stream cipher de-scrambler 310 and then de-serializer 312 de-serializes and aligns the data into 5-bit code groups. The 5-bit code groups are decoded into 4-bit data nibbles by a decoder 314. The start of stream delimiter is replaced with preamble nibbles and the end 30 of stream delimiter and idle codes are replaced with zeros. When an invalid code group is detected in the data stream, the Ethernet MAC 122 asserts the MII receive error RXER signal which asserts several error conditions which can be used to improperly terminate the data stream. While RXER is asserted, the receive data can be driven with "01" for an invalid data reception and "10" for a false carrier.

35 FIG. 9, shows controller portion of the Ethernet MAC. The MAC receiver routes nibble wide MII data through Cyclic redundancy checker 332 to a buffer 330 where it can be assembled into 32-bit long words. The data is forwarded to a 256 byte FIFO register 334 which is coupled to the ISB (not shown) by ISB interface 336. Ethernet packet data is moved from the FIFO 334

1 to system memory by the system DMA controller 150 (not shown). The MAC transmitter contains a 256 byte FIFO register 338 which is coupled to the ISB (not shown) via the ISB interface 336. An Ethernet packet is can be moved from system memory (not shown) into the FIFO 338 by the system DMA controller (not shown). The data can be forwarded through data  
5 buffer 340 to the MII (not shown) by a transmit state machine (not shown). The transmit state machine initiates transmission of the packet to the MII (not shown) when a FIFO watermark is reached or an end of packet is received. The transmit state machine also ensures that the correct inter packet gap (IPG) duration has expired before initiating packet transmission to the MII (not shown) from the data buffer 340.

10 The preamble and start of frame delimiter (SFD) can be initially transmitted over the MII, followed by transmission of actual data. If the packet is less than 64 bytes long, all zero pad bytes can be appended to the frame. CRC generator 342 automatically generates CRC which is appended if the packet is less than 64 bytes long.

#### 5. Universal Serial Bus Transceiver and MAC

15 The exemplary network gateway preferably includes a USB 1.1 compliant full speed (12 M b/sec) device interface. The USB 1.1 specification defines an industry-standard USB. The specification describes the bus attributes, the protocol definition, types of transactions, bus management, and the programming interface required to design and build systems and peripherals that are compliant with this standard. The USB 1.1 specification is incorporated herein by  
20 reference as if set forth in full. The (USB) can provide a ubiquitous link that can be used across a wide range of PC-to-telephone interconnects.

The USB interface, in the described embodiment, supports sixteen configurations, four concurrent interfaces per configuration, four alternative interfaces per interface and six concurrent endpoints. An endpoint is a uniquely identifiable portion of a USB device that is the  
25 termination of a data path between the host (e.g. MIP core) and device. Each USB logical device is composed of a collection of independent endpoints. Each logical device has a unique address assigned by the system at device attachment time. Each endpoint on a device is assigned a unique device-determined identifier called the endpoint number. Each endpoint has a device-determined direction of data flow. The combination of the device address, endpoint number, and  
30 direction allows each endpoint to be uniquely referenced. Each endpoint is a simplex connection that supports data flow in one direction: either input (from device to host) or output (from host to device). An endpoint has characteristics that determine the type of transfer service required between the endpoint and the client software.

FIG. 10 shows the USB transceiver. The USB transceiver uses a differential output  
35 driver 370 to drive the USB data signal onto the USB cable in accordance with the USB 1.1 specification. The driver can be a CMOS driver with an impedance that is less than the resistance specified in USB 1.1 specification so that a discrete series resistor may be included to make up the balance as required. A differential input receiver 372 can accept the USB data signal. The

1 receiver sensitivity may comply with the USB 1.1 specification with an input sensitivity (VDI)  
of at least about 200mV when both differential data inputs are in the differential common. In  
addition to the differential receiver, the described exemplary USB transceiver may include a  
single-ended receiver 374,376 for each of the two data lines. The transceiver can support both  
5 full- and low-speed data rates such that any type of device may be coupled to these ports. When  
the transceiver is in full-speed mode it can use full-speed signaling conventions and edge rates.  
In low-speed it can use low-speed signaling conventions and edge rate. The described exemplary  
transceiver preferably utilizes two logical levels, J and K data states, to communicate differential  
data in the system. Differential signaling complies with the crossover voltage requirements of  
10 Section 7.1.2 of the USB 1.1 specification.

Figure 10A shows the USB MAC. The USB MAC includes a USB packet processor 380  
that supports bulk in/out endpoints with packet sizes of eight, sixteen, thirty two or sixty four  
bytes. The USB packet processor 380 fragments USB packets and reassembles them into higher  
level packets with automatic detection of zero length USB packets. Internal USB configuration  
15 RAM 381 allows flexible configuration of the USB interface. The configuration RAM 381 can  
provide on the order of at least about 512 bytes of storage space for USB configuration  
descriptors. At power up of the network gateway, the MIPS core programs the configuration  
RAM 381 with a USB descriptor sequence. The MIPS core reads the values back from the  
configuration RAM to verify the accuracy of the descriptors in RAM. The MIPS core enables  
20 USB operation.

The USB MAC can support multiple configurations, interfaces, alternative interfaces and  
endpoints. The size of the configuration RAM will limit on the total length of the descriptor  
sequence. Therefore, the size of the configuration RAM may be optimized for integration into  
various applications which may require more than the described 512 bytes of storage space.  
25 There are five descriptors, the device descriptor, configuration descriptor, string descriptor, as  
well as interface and endpoint descriptors. A descriptor sequence contains one device descriptor  
and one or more configurations, interface and endpoint descriptors. The length of a  
device/configuration interface descriptor is nine bytes and the length of an endpoint descriptor  
is seven bytes. String descriptors are not stored in the configuration RAM.

30 The USB MAC processes all required USB standard device requests in hardware,  
including the clear\_feature, get\_configuration, get\_descriptor, get\_interface, get\_status, set  
address, set\_configuration, set\_feature, and set\_interface commands. The USB MAC extracts  
optional and class/vendor commands such as, for example, set\_descriptor, synch\_frame, and  
get\_string\_descriptor and forwards those commands to the MIPS core for processing. Upon  
35 receipt of such device requests the USB MAC preferably issues an interrupt to the MIPS core  
which then extracts the setup command and processes it.

In the described exemplary embodiment of the network gateway, there are preferably six  
physical USB endpoints: one bulk RX endpoint, one bulk TX endpoint, one isochronous RX

1 endpoint, one isochronous TX endpoint, and one interrupt RX endpoints. An isochronous transfer can provide guaranteed access to USB bandwidth with bounded latency and guaranteed constant data rate through the pipe as long as data is provided to the pipe. The endpoint can be a given configuration specifies the maximum size data payload that it can transmit or receive.

5 The USB uses this information during configuration to ensure that there is sufficient bus time to accommodate this maximum data payload in each frame. If there insufficient bus time for the maximum data payload, the configuration is established; if not, the configuration is not established. The interrupt transfer type supports those devices that send or receive small amounts of data infrequently, but with bounded service periods. An interrupt transfer preferably

10 provides a guaranteed maximum service period for the pipe as well as an attempt to re-transfer the data at the next period, in the case of occasional delivery failure due to error on the bus. The endpoint description identifies whether a given interrupt pipe's communication flow is into or out of the host.

Bulk transfers can support the exchange of relatively large amounts of data at highly

15 variable times where the transfer can use any available bandwidth. Bulk transfers preferably provide access to the USB on a bandwidth-available basis, with guaranteed delivery of data, but no guarantee of bandwidth or latency. In addition bulk transfers preferably attempts to re-transmit in the case of delivery failure. The bulk endpoint specifies the maximum data payload size that the endpoint can accept from or transmit to the bus. This maximum applies to the data payloads

20 of the data packets; i.e., the size specified is for the data field of the packet not including other protocol-required information. The bulk endpoint is designed to support a maximum data payload size. The bulk endpoint preferably reports in its configuration information the value for its maximum data payload size. The USB does not require that data payloads transmitted be exactly the maximum size i.e., if a data payload is less than the maximum, it does not need to be

25 padded to the maximum size. In the described exemplary embodiment, of the USB both RX/TX bulk endpoints can support a maximum USB packet size of eight, sixteen, thirty two, or sixty four bytes. Both RX/TX isochronous endpoints 382, 383 can support a maximum USB packet size of eight, sixteen, thirty two, sixty four, one hundred and twenty eight, two hundred and fifty six or five hundred and twelve bytes. The control endpoints can support a maximum packet size

30 of thirty two bytes and the interrupt RX interrupt endpoint can support a maximum USB packet size of eight bytes.

Both the bulk and isochronous endpoints 382, 383 can support in hardware the fragmentation of higher layer packets (such as Ethernet packets) into USB packets in the transmit direction and the reassembly of higher layer packets from USB packets in the receive direction.

35 An end of packet flag can be used to signal when a USB packet is shorter than the maximum packet size defined by the endpoint. In the case that the length of the higher layer packet is exactly an integer multiple of the maximum USB packet size, a zero length packet can be inserted to signal the end of packet. The USB MAC supports the generation of zero length packets in the

1 transmit direction and the detection of zero length packets in the receive direction.

The USB MAC may include internal RX and TX RAM 384, 385 for temporary data buffering among the bulk, isochronous and control endpoints. The endpoints are then forwarded to system memory. The USB preferably includes four direct memory access (DMA) channels  
5 for fast access to and from system memory through a system bus interface 386 coupled to the ISB. Preferably, two DMA channels are used for bulk RX/TX endpoints and two are used for isochronous RX/TX endpoints.

#### 6. Audio Processor

The audio processor module provides an interface between the voice processor and  
10 external subscriber line circuits (SLICs). Referring to FIG. 3, the audio processor includes an analog front end 172 which provides bi-directional exchange of signals over the voice bandwidth, including voice or fax/modem data signals modulated with a voice band carrier. The analog front end 172 can support four separate voice channels with an analog front end 172 having four pairs of 14-bit analog to digital converters (ADCs) and digital to analog converters (DACs).

FIG. 11 shows a block diagram of the analog front end of the audio processor. The digital  
15 input/output data of the DAC/ADC can be interpolated / decimated in the codec DSP logic block to provide 14-bit, 8 kHz input/output data for the audio processor 170. A pair of resistors 391a, 391b at the output of each DAC 390 converts the current output signal to a voltage. A pair of switches 393 can be provided between the output of the DAC and the input of the ADC to  
20 provide analog loopback test capability.

The analog front end may include a common mode voltage level generator 394 which provides an external common mode voltage level. Passive external circuitry coupled with the CM level generator 394 can be used to establish the DC level of the AC coupled input signals for the ADCs 392. A voltage reference 396 can be used to provide signals and bias currents for  
25 each of the ADC / DAC pairs and provide a bias current for the CM level generator 394. The reference may be overdriven with an external reference or may be left disconnected externally, allowing the internal bandgap to set the reference voltage. A clock generator 398 can be used to divide the 98.304 MHz PLL clock down to 49.152 MHz, 24.576 MHz and 12.288 MHz. The clock generator 398 provides a sample clock for the ADC 390 and DAC 392.

The external analog interface between each channel of the audio analog front end and an  
30 external SLIC is shown in FIG. 11A. In the described exemplary embodiment, of the analog front end resistors 391a and 391b convert the current output signal of DAC 390 to a voltage signal. Capacitors 402a, 402b and 402c provide low pass smoothing and anti-alias filtering of the attenuated signal. Op-amp 404 provides signal ended differential conversion and  
35 amplification of the DAC output which can then be forwarded to the SLIC 406. In the transmit direction, an RC network at the input of the ADC 392 provides balanced impedances at both ADC input pin and provide attenuation of the transmit signal at the positive input. The balanced impedance interface ensures that power supply and digital substrate noise affect both ADC inputs



1 equally. The ADC 392 samples the difference between the voltages at the inputs 408, 410 so that common noise can be rejected. The passive components fix the ADC input 410 at a constant DC level equal to the common mode level 412 output by the CM generator 394. The ADC input 408 varies in direct proportion to the transmit signal 414 from the SLIC 406.

5 Referring to FIG. 12 a voice synchronizer 419 synchronizes the near end voice samples received by the ADC 392 to the average timing regeneration circuit clock to ensure voice exchange between the network gateway and CMTS does not introduce jitter do to data underflow or overflow. A decimator 420 downsamples the synchronized output of the voice synchronizer 419 from 96 kHz to a 8 kHz rate. The decimator 420 can be implemented with a FIR filter which  
10 may be either minimum phase or linear phase. A minimum phase filter is preferred for application such as voice which need low group delay but may tolerate group delay distortion introduced by the minimum phase filter. The linear phase filter is preferred for fax and data modem applications. The decimator can be used to attenuate signals which can not be represented by the lower sampling rate, and then drop eleven out of every twelve samples. If a sample is not  
15 used at the output, there is no need to calculate its value. Therefore the FIR realization of the decimator can be set up to so that only the one sample out of twelve that is needed is calculated, and then the data set is advanced by twelve samples and the next output is calculated, thereby forgoing the intervening eleven samples.

The decimated signal can be filtered by a 60 Hz high pass filter 422. The filter 422  
20 reduces the 60 Hz interference induced on the transmit line due to proximity to power lines. The filter provides on the order of about 20 dB attenuation at 60 Hz relative to the transmission path loss at 1004 Hz. The filtered signal can be conditioned by a network echo canceller 424. The network echo canceller may interface with ITU-T G711, G726, G728 and G729 encoders. Double talk logic 423 can provide control functions including voice detection, automatic signal  
25 level control, transmit/receive gain switching balance, noise level estimation and double talk detection. The adaptive cancellation processes the receive side signal samples through an adaptive model of the hybrid coupling and subtracts the result from the transmit side in difference operator 426. A high pass filter 425 is coupled to the echo canceller input to match the echo path impulse response.

30 Programmable gain adjuster 428 applies a fixed gain or attenuation to output 426(a) of the difference operator 426. Gain adjuster 428 can provide programmable gain / attenuation adjustments of +/- 20 dB with step size of 1 dB. A fixed gain of attenuation 429 is applied to the gain adjusted signal. A multiplexer 427 coupled to the output of the fixed gain 429 allows the signal to be routed to a A-law /  $\mu$ -law (G.711 compatible) encoder 430 which is coupled to  
35 an external PCM port which allows for external testing of the audio processor module. Multiplexer 429 also forwards the gain adjusted output signal to the voice processor 160 via the DSP interface 168 (see FIG. 3).

The described exemplary embodiment of the audio processor includes multiplexer 431

1 coupled to the data interface in the receive mode. Multiplexer 431 may couple decoded samples  
to a A-law /  $\mu$ -law decoder 432 which is also coupled to an external PCM port to allow for  
external testing of the audio processor module. The multiplexer 431 may also forward decoded  
5 samples from the data interface to a gain adjuster 435 which applies a fixed gain or attenuation  
to the decoded signal 433. Gain adjuster 435 compensates for system level gain adjustments and  
may provide programmable gain/attenuation adjustments on the order of about +/- 20dB with a  
step size of 1 dB. A 1 kHz test tone generator 434 that provides a digital representation of a 1004  
Hz test tone at a level of 0 dBm. The test tone may be optionally injected into the data stream  
10 by summer 436 to debug and verify the audio processor. The test tone may be configurable in  
both frequency and amplitude, although it is preferably limited by the 8 kHz sample rate such that  
only 0-4 kHz may be allowed.

An interpolater 438 modifies the sample rate from 8 to 96 kHz. The interpolater 438 can  
be implemented with a FIR filter which may be either minimum phase or linear phase. A  
15 minimum phase filter is preferred for voice applications which require low group delay but may  
tolerate group delay distortion which may be introduced by the minimum phase filter. A linear  
phase filter is preferred for fax and or modem applications. In addition, a metering pulse  
generator 440 can be used to generate 12/16 kHz metering pulses that are summed with the  
interpolated signal by summer 442. The metering pulses allow a user to monitor the cost of the  
20 call as it progresses. The rate at which the metering pulse generator 440 transmits the metering  
pulses varies depending on the rate per minute of the call. The minimum gap between pulses is,  
by way of example, on the order of about 100 msec but may be much greater for inexpensive  
local calls. The amplitude of the metered pulses can be adjustable to accommodate impedance  
changes during the on hook, off hook and ringing states. The interpolated signals are forwarded  
to the DAC 390 for communication to a telephony device via the SLIC (not shown).

25 Power efficiency is often important for telephony applications. Therefore, the described  
exemplary embodiment of the audio processor includes the capability to enter a power  
saving/sleep mode wherein only those functions that are required for wake up are active. All  
circuits should be capable of reaching active steady state within about a 10 msec activation  
period.

30 The described exemplary embodiment of the preferred audio processor 170 further  
includes a ring generator which synthesizes reference waveforms which are forwarded to the  
SLIC to ring telephony devices. The ring generator can be used to support sinusoidal, trapezoidal  
and square waves. In addition the ring generator frequency, amplitude and DC offset can be  
configurable. A block diagram of a preferred ring generator 450 is shown in FIG. 13. The ring  
35 generator 450 includes a digital oscillator 452 which generates a sinusoid of a given frequency  
and amplitude determined by instructions 454, 456 from the DSP core of the voice processor 160  
via the DSP/PB interface 168 (see FIG. 3). The sample rate of the sinusoid can be, by way of  
example, on the order of about 1000 Hz, divided down from the 24.576 MHz system clock input

1 458. A variable clipper 460 symmetrically clips the sinusoid about zero such that the sinusoid  
may be converted into either a trapezoid or into a square wave. The DSP core of the voice  
processor 160 (see FIG. 3) can be used to define the thresholds 462 with which the sinusoidal  
waveform is clipped. The clipped waveform can be scaled by multiplier 464, which applies a  
5 signal attenuation 466 defined by the voice processor 160. Summer 468 provides a configurable  
DC offset by adding a DC bias 470 as established by the voice processor. The offset may vary  
from negative full scale to positive full scale. A converter 472 can be used to convert the ring  
waveform 468(a) into a single bit representation. A single pole analog filter may be included  
on the output of the converter to reduce the quantization noise generated by the converter 472.  
10 The filtered signal is then forwarded the analog front end 172 for communication to a telephony  
device via the SLIC. In the described exemplary embodiment, of the audio processor each audio  
channel may be routed to a PCM port to allow for system level PCM testing. The PCM system  
tests, by way of example, can require compliance with ITU G.711 for A-law and  $\mu$ -law encoding  
/ decoding.

#### 15 A. Voice Synchronization

Digitizing and transmitting voice data via packets in a network system is a common  
telephony problem. Customarily Pulse Code Modulation (PCM) techniques digitize a voice  
signal by sampling an analog voice signal and converting each sample into a digital code which  
is communicated between compatible receiving and transmitting systems on the near and far ends  
20 of the network. In addition, in a voice band data mode, the exemplary network gateway may  
transparently exchange data without modification (other than packetization) between a near end  
telephony device (or circuit switched network) and the packet based network. This is typically  
used for the exchange of fax and modem data when bandwidth concerns are minimal. The  
problem that arises is that the receiving system's clock may not be correlated with the  
25 transmitter's clock. This difference, even if minute, may cause the sample buffer in the receiving  
unit to underflow or overflow. In the case of data underflow, samples are extracted from a  
sample buffer faster than samples are written to the buffer so that the system may collapse from  
data starvation. During data overflow, voice signals transmitted from one communication port  
enter the sample buffer in the receiving unit faster than they are extracted. The resulting overflow  
30 of data signals may result in artifacts in a voice call or data in voiceband data mode.

To prevent data signal overflow and underflow, it is, therefore, desirable to synchronize  
the receiving clock to the incoming data rate. A voice synchronizer may be used for this purpose.  
Although the the voice synchronizer is described in the context of an audio processor system  
within a network gateway, those skilled in the art will appreciate that the voice synchronizer is  
35 likewise suitable for various other telephony and telecommunications application.

Referring to FIG. 14, network gateway 490 supports the exchange of voice between a  
hybrid fiber coaxial (HFC) network 492 and a traditional circuit switched 494 or a packet based  
network 496. In an exemplary embodiment, telephony device 500 is connected to the PSTN over

1 PSTN telephone gateway 502. The PSTN telephone gateway 502 may be clocked by a telephony  
network clock signal 504(a) from network clock reference 504 which is also coupled to CMTS  
506 such that the PSTN telephone gateway 502 may be synchronized with the CMTS clock for  
the transfer of PCM voice packets 502(a) between the CMTS 506 and the PSTN telephone  
5 gateway 502. The telephony network clock is preferably a conventional Building Integrated  
Timing Supply (BITS) clock. The equipment requirements for interfacing to this clock are  
known to those skilled in the art and are described in Bellcore document TR-NWT-001244 the  
content of which is incorporated herein by reference as if set forth in full. The CMTS clock is  
synchronized with the telephony network clock signal 504(b) via CMTS clock synchronizer 508  
10 which utilizes headend reference tick clock 510, as described in U.S. Pat. Application No.  
09/501,850, the contents of which is incorporated herein by reference as if set forth in full.

A timebase message generator 512 generates time messages or stamps which provide the  
network gateway 490 an absolute time reference which is representative of the frequency of the  
crystal oscillator timing reference (not shown) of the CMTS 506, which is in turn synchronized  
15 to the telephone network clock 504.

Referring to FIG. 14A, the network gateway 490 preferably includes a digital tracking  
loop 514 which provides a substantially stable clock output 520(a). A difference between the  
time stamps forwarded by the CMTS and the output of a local time reference 516, which is  
derived from a numerically controlled oscillator 518, is formed by differencing circuit 520. This  
20 difference defines a frequency error value. This frequency error value is filtered by loop  
averaging filter 521 which prevents undesirable deviations in the frequency error value from  
affecting the numerically controlled oscillator 518 in a manner which would decrease the stability  
thereof or cause the numerically controlled oscillator 518 to operate at other than the desired  
frequency. The loop filter 521 can be configured so as to facilitate the rapid acquisition of the  
25 frequency error value, despite the frequency error value being large, and then to reject  
comparatively large frequency error values as the digital tracking loop 514 converges, i.e., as the  
output of the local timing reference becomes nearly equal to the absolute time reference, thereby  
causing the frequency error value to approach zero.

Referring to FIG. 15, in the downstream direction, voice or voiceband data packets are  
30 received by the network gateway 490, demodulated and re-formatted by the downstream  
demodulator / DOCSIS MAC 522 and forwarded to system memory 524. The voice data is  
forwarded to a decoder system (not shown) within the voice processor 526. The decoder system  
preferably provides delay compensation, voice decoding, DTMF generation, call progress tone  
generation, comfort noise generation and lost frame recovery. Processed voice sample are then  
35 forwarded to a first voice queue 528 within the audio processor 170. A voice synchronizer 530  
is coupled to the output of the first voice queue 528. The voice synchronizer 530 re-samples the  
voice frames stored in the first voice queue 528 in accordance with an error signal and forwards  
re-sampled voice signals to a second voice queue 532 so that the rate at which samples are

1 removed from the second voice queue 532 by a DAC 534 matches the rate at which they are inserted into the second voice queue 532 by the voice synchronizer 530.

5 In operation, each time the clock of the DAC 534 decrements to zero, a sample can be removed from the second voice queue 532 and transmitted to the near end telephony device 536 via a subscriber line interface circuit 535. In the described exemplary embodiment, the DAC 534 is preferably driven by sampled DAC clock 538. In a jitter-free system, the DAC 534 removes frames from the second voice queue 532 at the exact same rate at which frames are inserted into the first voice queue 528 by the voice processor 526. However, when jitter or other impairments are present voice synchronization is needed because the DAC clock of the receive unit within the network gateway may not be correlated to the far end sample clock that generated the transmitted data.. In a data underflow condition in the downstream direction, the DAC clock 538 in the network gateway 490 leads the far end sample clock so that if left uncorrected samples would be removed from the second voice queue 532 faster than they are being inserted in the first voice queue 526. During a data overflow condition in the downstream direction, the DAC clock 538 in the network gateway lags the transmit clock in the far end sample clock so that samples are inserted into the voice queue faster than they are removed.

10 In the described exemplary network gateway, in the downstream direction a lost frame recovery engine in the voice and data processor is implemented whereby missing voice is filled with synthesized voice during data underflow conditions using the linear predictive coding model of speech. The voice is modelled using the pitch and spectral information from digital voice samples received prior to the lost packets. Similarly, during data overflow the voice and data processor preferably performs frame deletions to substantially eliminate the overflow condition. However, in voiceband data mode lost data may not be readily tolerated or recreated. Therefore, in voiceband data mode, the described exemplary voice synchronizer fractionally decimates the received voice signal stored in the first voice queue 526 during data overflow and fractionally interpolates the voice samples during data underflow. Although voice synchronization is described in the context of an audio processor for voice and voice band data exchange over cable modem, those skilled in the art will appreciate that the techniques described for signal synchronization are likewise suitable for various applications requiring the synchronization of a signal in a system having two uncorrelated clocks. Accordingly, the described exemplary embodiment for voice and voiceband data synchronization in a signal processing system is by way of example only and not by way of limitation.

20 For example, referring to FIG. 16 a given input voice signal 526 may be represented by a series of samples 526(a-i) shown with arrow terminations. In the described exemplary embodiment, samples 526(a-i) satisfy Nyquist sampling criteria so that input voice signal 526 may be completely reconstructed from the series of samples 526(a-i). Therefore, the input voice signal may be over sampled as represented by the dashed lines 528. For the data underflow case where the receive clock leads the transmit clock the input voice signal 526 may be sampled at a

1 slightly higher frequency 528(a-i) shown with circles, so as to substantially eliminate the data underflow condition, by generating additional samples as a function of time.

Similarly, for the data overflow condition, the input voice signal may be downsampled with a slightly lower frequency, shown with "x" terminations, so as to substantially eliminate the data overflow condition by generating fewer samples as a function of time. For example if the TRC clock 528 were 96.0000152587890625 kHz, and the transmit clock as represented by the rate at which samples are written into the first voice queue 520 is 96.0 kHz, then the voice synchronizer 522 would preferably generate an additional data sample every  $1/(0.0000152587890625) = 65536$  samples. Therefore, the incoming signal would have to be resampled at the rate of  $(65536)(96 \text{ kHz})$  or 6.291456 GHz. Advantageously, the described exemplary embodiment generates an error signal 522(a) as a function of the ratio between the rate at which frames arrive or are placed into the first voice queue 520 and the rate at which frames are removed from the second voice queue 524 so that only those samples that are required to fractionally decimate or interpolate the input voice signal should be computed.

Referring back to FIG. 15, in the upstream direction, the near end telephony device 536 transmits an analog voice signal to the network gateway 490 via the SLIC 535. An analog to digital converter 537 converts the incoming analog signal to digital samples with an ADC sample clock. If the TRC clock and the ADC clock are correlated; the ADC output is forwarded to the audio processor 537 for decimation, echo cancellation and other processing as previously described, prior to being forwarded to the encoder system of the voice processor 526. The voice processor encoder system can be provide, among other things, DTMF and call progress tone detection, voice activity detection, voice compression, and comfort noise estimation. Encoded voice sample are then forwarded to system memory 526 for transmission to the far end telephony device 500 by the upstream modulator and MAC 521. However, if the ADC 537 clock and the TRC clock are uncorrelated, voice synchronizer 530 should synchronize the digital voice samples to the TRC clock.

A general block diagram of an exemplary voice synchronizer 530 is shown in FIG. 17. The voice synchronizer preferably includes an error generation unit 552, a sampling rate tracker 554 and a voice queue or FIFO 555. The error generation unit 552 includes two counters each of which may be clocked by a locally generated reference at 24.576 MHz 553. In the upstream direction the ADC forwards digital samples to the sample rate tracker 554. With the assumption that the ADC clock and the TRC clock are uncorrelated, the ADC 8 kHz clock 556 drives a first counter 557 and the TRC 8 kHz clock 558 drives a second counter 559. Counter outputs 557(a) and 559(a) are forwarded to low pass filters 563, 563' that average the output count (see FIG. 18A). The sampling rate tracker 554 preferably fractionally decimates or interpolates the incoming signal 554(a) in accordance with the ratio of count A to count B. Thus if count A is larger than count B the sample rate tracker up-samples the incoming signal 554(a) by the ratio count A to count B. Otherwise the sample rate tracker 554 downsamples the incoming signal

554(a) by the ratio count A to count B. The sampling rate tracker 554 forwards the resampled signal 554(b) to the FIFO 555 wherein the samples are assembled into a frame. The assembled frame is then forwarded to an encoder system of the voice and data processor (not shown) for voice compression such as for example G.729.

Referring to FIG. 17B, the exemplary sample tracker operation may be best demonstrated by an exemplary. In FIG. 17B it is assumed that the input sample count shown as samples 1, 2, 3, and 4 is 65537 and the output sample count, shown as samples A, B, C and D, is 65536 thereby creating a slight overflow condition. The distance between for example, sample 2 and B is equal to  $1/65536$  of one input sample period. Therefore, an incoming 8 kHz signal would have to be resampled at the rate of  $(65536)(8 \text{ kHz})$  or 524.288 MHz to generate the additional samples required to eliminate the overflow condition. Referring to FIG. 17C, in the extreme 65535 zeros would be placed between each individual data sample of the input signal stored in a data buffer. In a typical application eighteen data points would be used to compute the resampling. The number of data points represents a trade between the quality of the conversion and computational complexity. Low pass filter coefficients (Coef-1 through Coef-1179648) would be computed which, when applied to the samples stored in the buffer yield the resampled signal.

Advantageously, the generation of the error signal, defined as the output count divided by the input count, facilitates the regeneration of the desired resampled output with a reduced number of filter coefficients. For example, referring to FIG. 17D, to compute output B we shift 655357 inputs with 65535 zeros, so that sample 65536 is not equal to zero and sample number 65537 is equal to zero. Therefore, only eighteen filter coefficients multiply non zero data and only those eighteen coefficients need be calculated for each sample. They are 2,  $2 + 65536$ ,  $2 + (2*65536)$ , ...  $2 + (17*65536)$ . The process would then be repeated to compute output sample C. The first 65534 samples are zeros, sample number 65536 is non zero and sample 65537 is equal to zero.

In the downstream direction incoming PCM voice data is forwarded to the sample rate tracker 554. The voice data may not be synchronized to the local DAC sample clock that converts the received voice samples to an analog format prior to transmission to the near end telephony device. In the downstream direction the first counter 557 is driven by an adjustable frame clock 560. The speed of the frame clock is negotiated between the network gateway and the CMTS and is assumed to be 10 msec. The frame clock 560 is generated by the MIPS core as a function of frame arrival time. The second counter 558 is preferably driven by a 100 Hz DAC clock 561 that is generated by a clock division counter as shown in FIG. 17A. The sampling rate tracker 554 preferably fractionally decimates or interpolates the incoming signal 554(a) in accordance with the ratio of count A to count B. Thus if count A is larger than count B the sample rate tracker up-samples the incoming signal 554(a) by the ratio count A to count B. Otherwise the sample rate tracker 554 downsamples the incoming signal 554(a) by the ratio

1 count A to count B. The sampling rate tracker 554 forwards the resampled signal 554(b) to the  
FIFO 555. The resampled signal is then converted to an analog signal via a DAC that may use  
24.576 MHz DAC clock.

5 FIG. 18 shows an alternative approach to voice synchronization. In the upstream  
direction the ADC 537 again digitizes analog voice data received from the near end telephony  
device 536 via the SLIC 535. The ADC 537 then forwards the digital samples to the sample rate  
tracker 554. In this instance the error generation unit utilizes a single counter to derive the offset  
between the ADC and TRC clocks. With this approach, the TRC 8 kHz clock 558 drives a  
counter 562 which utilizes an ADC 24.576 MHz clock 551 as a reference to count the number  
10 of periods within one 8 kHz TRC cycle. The counter forwards this count 562(a) to the low pass  
filter 563. The low pass filter as shown in FIG. 18A is preferably a single pole 564 filter that  
smooths the transitions between different sampling rates. Scale factor (b) 565 applies a fixed  
gain to the count out signal and scale factor (a) 566 is the time constant of the filter. Preferably  
the scale factors are interrelated according to the following:  $b = 1 - a$ . The length of the time  
15 constant represents a tradeoff between tracking accuracy and jitter and is preferably in the range  
of about 1-20 msec.

The sampling rate tracker 554 preferably fractionally decimates or interpolates the  
incoming signal 554(a) in accordance with the filtered counter output 563(a) as compared to  
3072. In the case where the TRC clock and the ADC clock are perfectly correlated the counter  
20 would output 3072 and the incoming signal 554(a) would not be resampled. However, for the  
case where the filtered counter output 563(a) is greater than 3072, the incoming signal is  
upsampled by the filtered counter output 563(a) divided by 3072 ( $A/3072$ ). Similarly, when  
counter output 562(a) is less than 3072, the incoming signal is down-sampled by the filtered  
counter output 563(a) divided by 3072 ( $A/3072$ ).

25 Referring to FIG. 18B, in an alternate voice synchronizer in the downstream direction an,  
the MIPS core 540 increments a counter 567 when the network gateway 490 receives a new voice  
frame from the CMTS 506. The counter 567 preferably utilizes a DAC 24.576 MHz clock 568  
as a reference. The counter output 567(a) provides the difference between the frame clock and  
the DAC clock. A low pass filter 563' again smooths or averages the difference in various  
30 frame arrival rates as previously described. The sampling rate tracker 554 preferably fractionally  
decimates or interpolates the incoming signal 530(a) in accordance with the output of filtered  
counter count 563(a) as compared to 24576. In the case where the frame arrival clock and the  
DAC clock are perfectly correlated the counter 567 would output 24576 and the incoming signal  
would not be resampled. However, for the case where the filtered counter output 563(a) is  
35 greater than 24576 the incoming signal is upsampled by the output 566(a) of the low pass filter  
divided by 24576 (i.e.  $A/24576$ ). Similarly, when the filtered counter output 563(a) is less than  
24576, the incoming signal is down-sampled by the output 563(a) of the low pass filter divided  
by 24576 ( $A/24576$ ).



## 2. Echo Cancellation

Referring to FIG. 19, in the analog front end of the audio processor, a DAC 570 forwards decoded voice signals to a local telephony device 571 and a ADC 572 receives signals from a local telephony device via a SLIC 573. The SLIC 573 contains a hybrid 574 which converts from two-four wires. Ideally the hybrid gain would be zero to eliminate coupling between the far end and near end transmission paths. However, the operation of the hybrid 574 typically results in a frequency dependent gain of less than about one-half. Hybrid gain commonly results in a telephony problem known as line echos, such as the reflection of the talker's voice at the mismatch created by the two-four wire conversion. If there is low end-to-end delay, echo from the far end is equivalent to side-tone (echo from the near-end), and therefore, not a problem. Side-tone gives users feedback as to how loud they are talking, and indeed, without side-tone, users tend to talk too loud. However, far end echo delays of more than about 10 to 30 msec significantly degrade the voice quality and are a major annoyance to the user. Therefore echo cancellers are typically used to remove far end echo.

Conventionally, a reference signal 575(a) is input into an adaptive filter 575 that models the transfer characteristics of the dialed telephone line (i.e. the hybrid plus the tail circuit which is the telephone line from DAC to ADC) so that the unwanted echo may preferably be canceled by subtracting filtered reference signal 575(b) from the near end digital input signal 572(a) in a difference operator 576. Typically, the adaptive filter 575 converges or adapts only in the absence of near end speech. Therefore, near end speech and/or noise present on the near end digital input signal 572(a), typically referred to as the double talk condition, may cause the adaptive filter 575 to diverge. Traditionally, echo cancellers utilize energy estimators 577a, 577b to estimate the energy ( $E_{near}$ ) of the near end signal 572(a) and the energy ( $E_{far}$ ) of the far end 575(a) signal. A typical double algorithm 576 then declares near end speech active, disabling adaptation of the adaptive filter 575, when the energy of the near end signal is greater than the energy of the far end signal times the hybrid gain(H), ( $E_{near} > H * E_{far}$ ).

A primary disadvantage of conventional approaches which utilize energy estimates is the delay introduced into the detection of near end speech by the energy estimators (typically low pass filters) which may significantly corrupt the output of difference operator 576, which is typically used as a the feedback error for filter adaptation. The described exemplary echo canceller includes a double talk algorithm that provides rapid detection of near end speech in the presence of far end speech along with a look ahead capability so that the adaptive filter may halt adaptation (i.e. freeze the filter taps or coefficients) before the near end speech reaches the difference operator.

Although echo cancellation is described in the context of an audio processor for voice exchange via a network gateway, those skilled in the art will appreciate that the techniques described for echo cancellation are likewise suitable for various applications requiring the cancellation of reflections, or other undesirable signals, from a transmission line. Accordingly,

the described exemplary embodiment for echo cancellation in a signal processing system is by way of example only and not by way of limitation.

Referring to FIG. 20, a high pass filter 587 receives a reference signal 587(a). The high pass filter 587 matches the echo path impulse response of the Rx data line. The output of the high pass filter 587 is input into the adaptive filter 575 that models the transfer characteristics of the dialed telephone line circuit. The adaptive filter 575 may be a linear transversal filter or other suitable finite impulse response filter. In addition, Rx data from the far end 581(a) is coupled to double talk logic 580 before the interpolator 581 of the audio processor (not shown) and the DAC 588 of the analog front end (not shown). The double talk logic 580 therefore receives a far end reference signal  $F(n)$  580(a) with an 8 kHz sampling rate. In addition, the double talk logic 580 is preferably coupled between the ADC 589 of the analog front end and the decimator 582 of the audio processor (not shown). A downsampler 583 performs 12:1 sample decimation of the 96kHz near end Tx data 572(a) and forwards the decimated near end data samples 583(a) to the double talk logic at an 8 kHz sample rate. To minimize delay, the downsampler does not low pass filter the near end samples 572(a) prior to decimation. Aliasing components which may be created are insignificant in that the output of the downsampler 583(a) simply drives the double talk detection logic 580 and is not transmitted to the far end. An energy estimator 584 estimates the background noise level of the decimated near end signal 583(a) and forwards the estimated level to the double talk logic 580. The energy estimator 584 is preferably a low pass filter with a long time constant, on the order of about 10 seconds. With a long time constant the energy estimator tends to track the minimum energy level of the decimated near end signal 583(a). Energy estimator 585 estimates the short term energy of the far end TX data  $F(n)$ .

The adaptive filter 575 can be based upon a normalized least mean square algorithm (NLMS) as described in S. Haykin, Adaptive Filter Theory, and T. Parsons, Voice and Speech Processing, the contents of which are incorporated herein by reference as if set forth in full. An error signal 576(a) at the output of the difference operator 576 for the filter adaptation may be characterized as follows:

$$e(n) = Tx(n) - \sum w(j)F(n-j)$$

where  $e(n)$  is the error signal at time  $n$ ,  $F(n)$  is the reference signal 587(a) at time  $n$  and  $Tx(n)$  is the Tx data signal 586(a) input into the difference operator 576 at time  $n$ , and  $w(j)$  are the coefficients of the transversal filter where the dimension of the transversal filter is the worst case echo path length (i.e. the length of the tail circuit  $L$ ) and  $W(j)$ , for  $j=0$  to  $L-1$ , is given by:

$$w(j) = w(j) + \mu * e(n) * F(n-j)$$

wherein  $w(j)$  is preferably initialized to a reasonable value such as for example zero.

Assuming a block size of four msec (or 32 samples at a sampling rate of 8 kHz), the short term average energy of the reference signal  $E_{ref}$  is the sum of the last 32 reference samples so that the convergence gain may be given by:

$$\mu = \frac{\alpha}{E_{ref}(n)}$$

1 where  $\alpha$  is the adaptation step size and  $E_{ref}$  is the energy estimate of the far end data sample  $F(n)$ . In the described exemplary embodiment  $\alpha$ , is set to zero when near end voice is detected so that the convergence gain  $\mu$  is equal to zero and the filter coefficients are not updated. Otherwise  $\alpha$  is set to a constant of less than one and preferably in the range of 0.8-0.95. One of skill in the art will appreciate that the adaptive filter may be implemented in a variety of ways, including fixed point rather than the described floating point realization. Accordingly, the described exemplary adaptation logic is by way of example only and not by way of limitation.

5 The 96 kHz near end Tx data samples 572(a) are also decimated by a second 12:1 decimator 582. However, decimator 582 does include a low pass filter capability to prevent aliasing of the decimated signal. The decimated output 582(a) is forwarded to a 60 Hz high pass filter 586 which reduces the 60 Hz interference induced on the transmit line due to proximity to power lines. Filtered output 586(a) is input to the difference operator 576 that preferably cancels unwanted echo by subtracting filtered reference signal 575(b) from the filter output signal 586(a).

10 In the described exemplary embodiment, the adaptive filter 575 models the transfer characteristics of the hybrid and the tail circuit of the telephone circuit. The tail length supported should preferably be at least 8 msec. The adaptive filter 575 may be a linear transversal filter or other suitable finite impulse response filter. The echo canceller preferably converges or adapts only in the absence of near end speech. Therefore, near end speech and/or noise present on the input signal 572(a) may cause the adaptive filter 575 to diverge. To avoid divergence, the adaptive filter 575 can be selectively enabled by the double talk logic 580. The double talk logic 580 utilizes a sample based algorithm to detect the presence of near end speech without incurring the delays associated with conventional systems in accordance with the following equation:

$$| \text{Near} | > H * \text{Peak}\{| F(n) | \} + \text{Background\_Noise}(n)$$

25 The double talk logic 580 is used to declare near end speech active when the absolute value of the decimated near end signal 583(a) is greater than the product of the hybrid gain (H) and a peak statistic of the far end data samples 581(a) summed with the estimated background noise of the transmit data samples. The hybrid gain is generally a constant preferably less than about one-half. The background noise for a typical voice channel is on the order of about -70 dBm which is far less than average active speech levels, which are typically in the order of about -25 dBm. The background noise estimate is therefore initialized to a value of about -70 dBm and thereafter periodically updated 584(a) by the energy estimator 584. The peak statistic of the far end data samples is defined by the following logic:

If  $\max\{A * [| F(n) |, \dots, | F(n-L-1) |]\} > \text{Peak}(n-1)$  then

$\text{Peak}(n) = \max\{A [| F(n) |, \dots, | F(n-L-1) |]\}$

else

$\text{Peak}(n) = d * \text{Peak}(n-1);$

35 where A is a weighting function that is greater than zero and less than or equal to one. The parameter L is the number samples over which the maximum is determined, typically in the range

1 of zero to one hundred and twenty eight samples and preferably on the order of about 64 samples.  
The parameter  $d$  is preferably a constant that is also greater than zero and less than or equal to  
one and preferably on the order of about 0.99. Therefore, to determined the peak statistic of the  
far end, the double talk logic applies a weighting factor  $A$  to the absolute value of the current  
5 sample  $(F(n))$  and previous  $L$  samples  $(F(n-L))$ . If the maximum product is greater than the  
previous peak statistic  $Peak(n-1)$  then the current peak statistic  $Peak(n)$  is set at the maximum  
of the product of the weighting factor and far end samples. Otherwise the current peak statistic  
 $Peak(n)$  is set equal to  $d$  times the value of the previous peak statistic  $Peak(n-1)$ .

10 In the described exemplary embodiment,  $A$ ,  $L$  and  $d$  are empirically determined to  
optimize performance and computational load for a given application. For example, double logic  
580 can more accurately detect near end speech if the maximum is determined over a larger  
number of samples  $L$ . However, computational intensity also increases with increasing number  
of samples  $L$ .  $A$  and  $d$  can be inversely related to the number of sample  $L$ , so that  $A$  and  $d$  are  
smaller for larger number of samples and vice versa.

15 In the described exemplary embodiment, there is a delay associated with the high-pass  
filter 586 and the decimator 582. The double talk logic 580, which has negligible delays, can  
receive and process near end data samples prior to their arrival at the difference operator 576.  
Thus, the delay associated with the high-pass filter 586 and the second decimator 582 provide  
a look-ahead of  $M$  samples allowing the double talk logic 580 to preferably disable adaptation  
20 of the adaptive filter 575  $M$  samples before the near-end signal reaches the difference operator  
56. The look ahead capability  $M$  is the sum of the equivalent delays associated with the high pass  
filter 586 and the second decimator 582 and is typically two-three 8kHz samples for a ITU-T  
G712 compliant system.

25 FIG. 20A shows another approach for echo cancellation where the near end digital signal  
after decimation to an 8 kHz signal 582(a) is input to the double talk logic 580. This approach  
can be utilized in systems where the echo canceller and codec are not integrated so that the near  
end data samples have previously been decimated. In this instance, a look ahead buffer 588  
receives, buffers, and forwards decimated near end signals 582(a) to the difference operator 576,  
providing a look ahead capability of  $M$  samples where  $M$  may be optimized for a given  
30 application to balance performance, computational intensity and delay.

The relative strength of the near end signal compared to the echo coupled through the  
hybrid increases with decreasing hybrid gain ( $H$ ) so that in the described exemplary embodiment,  
the accuracy of near end voice detection increases with decreasing hybrid gain( $H$ ). Referring to  
FIG. 21, in another aspect of the present invention, a short adaptive filter 590 is integrated into  
35 the preferred double talk detection algorithm. The adaptive filter 590 models the transfer  
characteristics of the dialed telephone line circuit. The adaptive filter 590 may be a linear  
transversal filter or other suitable finite impulse response filter. An error signal 591(a) at the  
output of the difference operator 591 for filter adaptation may be characterized as follows:

$$e_0(n) = Tx_0(n) - \sum_{j=0}^{K-1} w_0(j) F(n-j)$$

where  $e_0(n)$  is the error signal at time  $n$  591(a),  $F(n)$  is the reference signal 580(a) at time  $n$  and  $T_{x_0}(n)$  is the Tx data signal 591(b) input into difference operator 591 at time  $n$ , and  $w_0(j)$  are the coefficients of the transversal filter where the dimension of the transversal filter is preferably the worst case echo path length (i.e. the length of the tail circuit  $K$ ) and  $w_0(j)$ , for  $j=0$  to  $K-1$ , is given by:

$$w_0(j) = w_0(j) + \mu * e_0(n) * F(n-j)$$

wherein  $w_0(j)$  is preferably initialized to a reasonable value such as for example zero.

Assuming a block size of one msec (or 8 samples at a sampling rate of 8 kHz), the short term average energy of the reference signal  $E_{ref}$  is the sum of the last eight samples so that the convergence gain may be given by:

$$\mu = \frac{\alpha}{E_{ref}(n)}$$

where  $\alpha$  is the adaptation step size and  $E_{ref}$  is the energy estimate of the far end data sample  $F(n)$ . In the described exemplary embodiment, the double talk logic 580 does not selectively enable / disable adaptation of the filter 590 in accordance with the detection of near end speech so that filter 590 continuously adapts. Therefore, to reduce the computational burden placed upon the system and to prevent the filter from diverging the adaptive filter 590 can be figured to adapt very slowly so that  $\alpha$  is preferably in the range of about 0.01 - 0.0001.

The adaptive filter 590 again filters the far end reference signal 581(a) so that the echo level is can be reduced by subtracting filtered reference signal 590(b) from the Tx data samples 591(b) in a difference operator 591. The adaptive filter 590 can reduce line echos about 6-12 dB so as to improve the performance of the double talk logic. In the described exemplary embodiment, the output 591(a) of the difference operator 591, (i.e. Tx data samples with reduced echo) is then forwarded to double talk logic 580 which then detects near end speech and selectively enables/disables adaptation of the adaptive filter 575(see FIG. 20).

## 7. Voice Processor

The Internet is a loose association of thousands of networks and millions of computers across the world that are interconnected through communication links. The emergence of Internet Protocol (IP) as the standard transport protocol for packet based networks has enabled an on-line revolution in communications service and applications. Traditional dial-up modems provide online access through the public telephone network at up to 56 Kbps (equal to 56,000 bits per second). A cable modem, on the other hand, provides users with high-speed Internet access through a cable television network at data rates as high as 56 Mbps. However, traditional cable modem service has been limited to data applications so that the realization of diverse communications services at increased data rates requires the development of a common broadband cable access network with integrated voice and data services. Cable Television Laboratories, Inc. (CableLabs®) a membership organization consisting of cable television system operators developed PacketCable 1.0 which defines interface specifications for interoperable

1 equipment capable of providing packet-based voice, video and other high-speed multimedia services over hybrid fiber coax (HFC) cable systems utilizing the DOCSIS protocol.

5 The described exemplary network gateway includes a voice and data processor that supports the exchange of voice and data between a traditional circuit switched and a packet based network via a DOCSIS HFC network. The exemplary voice and data processor may be implemented with a programmable DSP software architecture as shown in FIG. 22. This architecture includes a high speed DSP 600 with program memory 602, preferably on the order of about a 80k word SRAM, and data memory 604 preferably on the order of about a 48k word SRAM. A PCM highway 606 provides the voice and data processor 160 access to the audio  
10 processor and optional external audio processing circuits. A grant synchronizer 608 insures delivery of samples to the network gateway for upstream transmission. The grant synchronizer signals the DSP 600 that a pending grant is about to arrive at the network gateway so as to allow the DSP 600 to synchronize itself to scheduled grants at the network gateway. A host interface 610 transfers data, control and status messages between the DSP 600 and the MIPS core 128.

15 The described exemplary embodiment preferably provides embedded media terminal adapter (MTA) capability in compliance with PacketCable 1.0. The exemplary embedded MTA may be implemented with the programmable DSP software architecture to provide a subscriber side interface to the subscriber's telephony device via the voice and data processor, as well as a network side interface to the DOCSIS cable modem. Referring to FIG. 23 the preferred  
20 embedded MTA 620 includes a host application programming interface (HAPI) 621 that provides a software messaging interface between the MIPS host and the voice and data processor DSP. The HAPI 621 facilitates the issuing of commands from the MIPS host to the voice and data processor DSP as well the sending of events from the DSP to the MIPS core host.

25 In addition, the MTA 620 can provide all signaling and encapsulation elements required to provide telephony service over a DOCSIS HFC network 622 including media transport and call signaling via quality service logic 623. For example, gateway control protocol (GCP) logic 624 receives and mediates call-signaling information between the PacketCable network and the PSTN. The GCP logic 624 maintains and controls the overall call state for calls requiring PSTN interconnection. The GCP logic 624 controls the voice and data processor 626, via the MTA 620  
30 and HAPI interface 621, by instructing it to create, modify, and delete connections that support the media stream over the IP network. The GCP logic 624 also instructs the voice and data processor to detect and generate events and signals. The GCP logic 624 also exercise attribute control over the voice and data processor 626 providing instructions as to which attributes to apply to a connection, such as, for example, encoding method, use of echo cancellation, security  
35 parameters, etc.

The GCP logic 624 also interfaces with an external control element called a call agent or call management server (CMS) 628 to terminate and generate the call signaling from and to the PacketCable side of the network in accordance with the network-based call signaling (NCS)

1 protocol specification. The PacketCable 1.0 NCS architecture places call state and feature  
implementation in the centralized CMS 628, and places telephony device controls in the MTA  
620. The MTA 620 passes device events to the CMS 628, and responds to commands issued  
from the CMS. The CMS, is responsible for setting up and tearing down calls, providing  
5 advanced services such as custom calling features, performing call authorization, and generating  
billing event records, etc. For example, the CMS 628 instructs the MTA 620 to inform the CMS  
628 when the phone goes off hook, and seven dual tone multi frequency (DTMF) digits have  
been entered. The CMS 628 instructs the MTA 620 to create a connection, reserve quality of  
service (QoS) resources through the access network for the pending voice connection, and to play  
10 a locally generated ringback tone. The CMS in turn communicates with a remote CMS (or MGC)  
to setup the call. When the CMS detects answer from the far end, it instructs the MTA to stop  
the ringback tone, activate the media connection between the MTA and the far-end MTA, and  
begin sending and receiving media stream packets.

When a voice channel is successfully established, real time transport protocol (RTP) is  
15 used to transport all media streams in a PacketCable compliant network to guarantee  
interoperability. Real time transport protocol (RTP) provides end-to-end delivery services for  
data with real time characteristics, such as interactive audio and video. Those services include  
payload type identification, sequence numbering, timestamping and delivery monitoring of the  
quality of service (QoS) and conveys to participants statistics such as for example packet and byte  
20 counts for the session. RTP resides right above the transport layer. The described exemplary  
embedded MTA 620 preferably includes RTP logic 630 that converts RTP packets (headers) to  
a protocol independent format utilized by the voice and data processor 626 and vice versa.

The described exemplary embedded MTA preferably includes channel associated  
signaling (CAS) logic 632 resident on the MIPS core that interfaces with the subscriber line  
25 interface circuits 634 via the GPIO interface 184 (see FIG. 3) to provide ring generation,  
hookswitch detection, and battery voltage control. The CAS logic 632 preferably supports  
custom calling features such as for exam distinctive ringing.

The described exemplary embedded MTA 620 preferably includes MTA device  
provisioning logic 636 which enables the embedded MTA 620 to register and provide subscriber  
30 services over the HFC network 622. Provisioning logic 636 provides initialization,  
authentication, and registration functions. The Provisioning logic 636 also provides attribute  
definitions required in the MTA configuration file. The provisioning logic 636 includes a SNMP  
logic 638 that exchanges device information and endpoint information between the MTA 620 and  
an external control element called a provisioning server (not shown). The MTA also sends  
35 notification to the provisioning server that provisioning has been completed along with a pass/fail  
status using the SNMP protocol.

The Provisioning logic 636 also includes DHCP logic 640 which interfaces with an  
external dynamic host configuration protocol (DHCP) server to assign an IP address to the MTA.

1 The DHCP server (not shown) is a back office network element used during the MTA device provisioning process to dynamically allocate IP addresses and other client configuration information. Further provisioning logic preferably includes domain name server (DNS) logic 642 which interfaces with an external DNS server(not shown) to obtain the IP address of a  
5 PacketCable server given its fully\qualified domain name.

The MTA configuration file is downloaded to the MTA from an external trivial file transfer protocol (TFTP) server (not shown) through TFTP logic 644. The TFTP server is a back office network element used during the MTA device provisioning process to download configuration files to the MTA. An HTTP Server may be used instead of a TFTP server to  
10 download configuration files to the MTA.

Each of PacketCable's protocol interfaces is subject to threats that could pose security risks to both the subscriber and service provider. The PacketCable architecture addresses these threats by specifying, for each defined protocol interface, the underlying security mechanisms (such as IPsec) that provide the protocol interface with the security services it requires, e.g.,  
15 authentication, integrity, confidentiality. Security logic 646 is PacketCable compliant and provides for voice and provides end-to-end encryption of RTP media streams and signaling messages, to reduce the threat of unauthorized interception of communications. The security logic 646 preferably provides additional security services such as, for example, authentication, access control, integrity, confidentiality and non-repudiation.

20 DOCSIS service logic 648 preferably provides the primary interface between the MTA 620 and the DOCSIS cable modem (i.e. DOCSIS MAC and modulator / demodulator) of the network gateway. The DOCSIS service logic 648 provides multiple sub-interfaces such as for example a control sub-interface which manages DOCSIS service-flows and associated QoS traffic parameters and classification rules as well as a synchronization interface which is used to  
25 synchronize packet and scheduling prioritization for minimization of latency and jitter with guaranteed minimum constant bit rate scheduling. In addition, the DOCSIS service logic is used to request bandwidth and QoS resources related to the bandwidth. The DOCSIS cable modem features of the network gateway then negotiate reserve bandwidth, guaranteed minimum bit rate etc, utilizing DOCSIS 1.1 quality of service feature. Similarly, DOCSIS service logic 648  
30 preferably includes a transport interface which is used to process packets in the media stream and perform appropriate per-packet QoS processing.

The exemplary embedded MTA may best be illustrated in the context of a typical voice communication across the DOCSIS HFC network. The user initiates a communication by going off hook. The CAS detects the off hook condition from the SLIC and sends an off hook event  
35 to the MTA call client. The MTA call client then instructs the GCP logic to generate a off hook signal. The GCP logic generates an of hook signal which is forwarded to the MTA call client and transmitted out the QoS service logic to the call management server via the DOCSIS MAC and upstream modulator of the network gateway and the CMTS. The call management server



1 typically would transmit a return signal via the CMTS, DOCSIS MAC and downstream  
demodulator of the network gateway to the MTA call client via the QoS service logic. The MTA  
call client preferably forwards that signal to the GCP logic which decodes the signal, typically  
play dial tone. The GCP logic would then signal the MTA call client to play dial tone. The MTA  
5 call client then sends a command to the voice and data processor via the HAPI interface to play  
dial tone. The user then hears a dial tone.

Upon hearing a dial tone a user will then typically dial a number. The voice and data  
processor includes a DTMF detector which detects the dialed digits and forwards the detected  
digits to the MTA call client as events via the HAPI interface. The MTA call client forwards the  
10 event to the GCP logic which encodes the dialed digits into a signaling message which is returned  
to the MTA call client. The MTA call client transmits the signaling message out the QoS service  
logic to the call management server via the DOCSIS MAC and upstream modulator of the  
network gateway and the CMTS. The call management server would then instruct a called party  
MTA to generate a ring to the called number. If the called number answers by going off hook,  
15 the CAS of the called MTA would detect an off hook condition and signal the call management  
server. The call management server then instructs the MTA call client via the CMTS, and  
downstream demodulator, DOCSIS MAC and QoS service logic of the network gateway to  
establish a voice connection with a given set of features, i.e. use echo cancellation, and silence  
suppression, use given coder etc. In addition, the MTA call client is given the IP address of the  
20 called party, to which the RTP voice packets should be sent. The MTA call client forwards the  
received message to the GCP logic which decodes the received message. The GCP logic  
generates attribute instructions for the voice and data processor such as, for example, encoding  
method, use of echo cancellation, security parameters, etc. which are communicated to the voice  
and data processor via the MTA call client and the HAPI interface.

25 Voice packets are then exchanged. For example, if the calling party speaks, the voice and  
data processor would process the voice and forward voice packets the MTA call client via the  
HAPI interface. The MTA call client would then forward those voice packet the RTP logic  
which would convert the packet from a protocol independent packet format to the RTP format.  
The RTP voice packets are then returned to the MTA which transmits the RTP voice packet to  
30 the CMTS via the QoS service logic and the DOCSIS MAC and upstream demodulator of the  
network gateway. The voice packets are then routed to the called party. Similarly, voice packets  
from the called party are communicated to the MTA of the call client via the QoS service logic.  
The MTA call client forwards the RTP voice packets to the RTP logic which converts the packet  
from the RTP format to the protocol independent packet format. The protocol independent voice  
35 packets are returned to the MTA call client which forwards them to the voice and data processor  
via the HAPI interface. The voice and data processor decodes the packets and communicates a  
digital stream to the called party. Voice exchange would continue in a similar manner until an  
on hook condition is detected by either the calling or called party CAS which would forwarded

1 a on hook detection event to its respective MTA. The MTA would instructs the GCP logic to  
generate a hook detection signaling message which is returned to the MTA and forwarded to the  
call management server. The call management server would generate a request to play (dial tone,  
silence or receiver off hook) which is forwarded to the opposite MTA. The MTA would forward  
5 the request to the GCP logic which would then instruct the voice and data processor to play dial  
tone via the MTA and HAPI interface.

Telephony calls in the other direction are similarly processed. For example, the call  
management server instructs the MTA called client to ring a dialed number. The MTA called  
client instructs the GCP logic to generates an command to ring the dialed number. The command  
10 is then forwarded to the CAS via the MTA called client. The CAS generates a ring signal and  
forwards that signal to the SLIC which then rings the called telephony device. The MTA called  
client may also instruct the GCP logic to present call ID which preferably generates a command  
for the voice and data processor to present caller ID. If the user picks up the phone the CAS  
would detect an off hook condition and signal an off hook event back to the MTA. The MTA  
15 called client would then instruct the GCP logic to create an off hook detection signaling message,  
which when created is returned to the MTA and forwarded to the external call management  
server via the QoS service logic, DOCSIS MAC and upstream modulator of the network gateway  
and the CMTS. A communication channel would again be established with a given set of  
attributes as previously described.

20 Referring to FIG. 24, the GCP logic implements the NCS features of the PacketCable 1.0  
specification including protocol transaction management, parsing, verification and formatting of  
protocol data units (PDUs) and other utility functions. The PacketCable protocol typically  
requires an acknowledgment that a request has been received. Therefore, the GCP logic includes  
transmitter queue 674 which registers all requests sent from the GCP logic to the call  
25 management server via a transmitter scheduler 672 and the MTA. A receive router 674 then  
analyzes packets received from the call management server via the MTA call client to determine  
if the received message is an acknowledgment or a new request. The receiver router 674  
forwards new requests to a transaction queue 676. If the received packet is an acknowledgment  
of a request previously sent from the GCP logic to the call management server, the receiver router  
30 674 clears that request that request by forwarding the acknowledgment 674(a) to the transmitter  
queue 670. The transmitter queue 670 periodically retransmits buffered requests until an  
acknowledgment from the call management server is received.

The transaction queue 676 registers the receipt of a given request from the call  
management server as well as the transmission of an acknowledgment of that request from the  
35 GCP logic to the call management server, so that the transaction queue 676 may preferably detect  
requests re-transmitted from the call management server for which an acknowledgment has  
already been sent. In addition, the transaction queue 676 parses the incoming requests according  
to the type of requests. The transaction queue 676 routes connection messages to a connection

1 message parser 678. Connections messages are used to create or terminate a connection. The  
connection message parser generates and forwards connection controls (including connection  
attributes) 678(a) to the MTA which are then communicated to the voice and data processor via  
the HAPI interface. The transaction queue 676 routes request / notify messages to a request /  
5 notify message parser 680. Request / notify messages may be signal requests such as play dial  
tone, busy tone etc. or requests to detect events such as for example, detect DTMF digits, fax  
tone, modem tone etc. The request / notify message parser 680 preferably generates and forwards  
signal requests 680(a) to the voice and data processor via the MTA and HAPI interface. The  
request / notify message parser 680 flags event detection requests 680(b) to an event filter 682.

10 Actual events detected by the voice and data processor (such as, for example, fax tone)  
or the CAS (such as, for example, off hook detected) are forwarded to the event filter via the  
MTA call client. The event filter 682 filters the events provided by the voice and data processor  
and CAS via the call client, and only transmits those detected events that the call management  
15 server requested, as indicted by flags 680(b) communicated to the event filter by the request /  
notify message parser 680. The event filter 682 preferably forwards detected events of interest  
to the call management server to a message formatter 684. The message formatter 684 formats  
the detected event into the appropriate protocol and forwards the detected event message to  
transmitter queue 670, which registers the message and will retransmit the message if an  
20 acknowledgment is not received in a timely manner via the receiver router 674(a). The  
transmitter queue 670 forwards the message to the transmitter scheduler 672 which bundles  
outgoing messages and forwards them to the MTA call client (not shown) for communication to  
the call management server.

25 The PacketCable 1.0 specification provides for the use of a digit map which is designed  
to reduce the number of messages communicated between the call management server and the  
MTA call agent when a user is dialing a number. For example, the dialing of long distance  
number involves the use of ten digits (i.e. the area code and number) which would require ten  
requests and acknowledgments i.e. one per digit. In the alternative the call management server  
may provide a digit map to the MTA call client which instruct digit map logic 686 to collect  
30 detected digits from the voice and data processor according to a specified format, for example  
ten digits for long distance call. The digit map logic 686 then forwards for example all ten digits  
to the event filter which filters the digit detection, and forwards events of interest to the message  
parser 684 for communication to the call management server as previously described through  
transmitter queue 670 and transmitter scheduler 672.

35 Event quarantine logic 688 buffers detected events received from the CAS or voice and  
data processor via the MTA call client for which the event filter has not received a detect event  
request from the call server manager via the request / notify message parser flag 680(b).  
Responses or the result 690(a) of a connection or signal requests are forwarded from the MTA

1 call client to a response formatter 690 with the GCP logic which formats the result into the proper  
protocol and forwards that result to the transmitter scheduler 672 for communication to the call  
management server via the MTA call client. In addition, the response formatter 690 notifies the  
transaction queue 676 that an acknowledgment has been sent in response to a given request. The  
5 transaction queue 676 may then detect the re-transmission of a request from the call management  
server should that acknowledgment be lost or otherwise not received by the call management  
server. The transaction queue 676 preferably instructs the response formatter 690 to retransmit  
an acknowledgment when the transaction queue 676 detects the re-transmission of a request for  
which an acknowledgment had been previously sent.

10 Referring to FIG. 25, RTP logic 630 preferably converts RTP packets to the protocol  
independent packet format utilized on the voice and data processor and vice versa. In the  
described exemplary embodiment, the protocol independent packet payload is preferably identical  
to the RTP packet payload so that the RTP logic 630 need only convert between RTP and  
xChange headers. In the described exemplary embodiment a RTP shim 700 provides two way  
15 exchange of protocol independent packets with the MTA call client 620. The RTP shim 700  
accepts these packets from the MTA call client 620, converts these packet header, to a RTP  
header and forwards the RTP packet to a RTP stack 702. The RTP stack 702 provides the  
statistical monitoring function required by the RTP protocol such as, for example, collecting  
statistics on the number of bytes sent and received, the number of packets lost and how much  
20 jitter was present. The RTP stack 702 provides two way exchange of RTP packet between the  
MTA call client and the RTP logic 630, receiving RTP packets from the RTP shim 700 and  
forwarding them to the MTA call client 620 and vice versa.

Referring to FIG. 26 the described exemplary host application programming interface  
(HAPI) provides a software messaging interface between the MTA call client 620 and the voice  
25 and data processor DSP. The HAPI 621 facilitates the issuing of commands from the MIPS  
core via the MTA call client to the voice and data processor DSP as well the sending of events  
from the DSP to the MTA call client 620. The HAPI interface 621 can be scalable to support  
multiple channels as well as multiple DSPs. In addition the HAPI 626 is a thread safe  
environment, providing multi-threading capability to support multiple channels. Multi-threading  
30 provides multiple, simultaneously-executing "threads" within a single application. The benefit  
of a multi-threaded application is similar to that of a multi-tasked operating system, the  
application may perform multiple tasks by using separate threads.

In the described exemplary HAPI, an application programming interface (API) 706  
receives commands from and issues responses to the MTA call client 620. The API 706  
35 processes incoming packets adding an additional header word to identify the channel which  
should process the packet as well as other processing parameters such as for example, packet type  
(i.e. voice or fax), packet length. The API forwards processed API messages 706(a) to a physical  
interface (firmware layer) 708 that interfaces with the host port of the DSP underlying the voice

1 and data processor so as to isolate the MTA call client software and the signal processing  
software from the underlying hardware. This methodology allows the software to be ported to  
various hardware platforms by porting only the hardware interface portions of the HAPI interface  
621 to the target hardware. The physical interface 708 formats the message in accordance with  
5 the underlying DSP and forwards or transmits the message to the telephony algorithms executing  
on the DSP.

Similarly, the underlying DSP forwards processed packets, such as, for example, encoded  
voice packets, to the physical interface 708. The physical interface 708 preferably reformats the  
response into an API message. When a processed packet is forwarded to the physical interface  
10 708 the underlying DSP also interrupts a APITask Thread 710 that retrieves the processed API  
messages from the physical interface 708. The APITask Thread 710 determines whether the API  
message is an event such as, for example, a voice or fax packet or a DTMF detection which are  
forwarded directly to the MTA call client 710(a), or a response to a command/request from the  
MTA call client. For example, the MTA call client may command the voice and data processor  
15 to turn off the echo canceller. Such a command is preferably processed by the API interface 706  
to add the appropriate header word and forwarded to the physical interface 708. The physical  
interface 708 formats the message in accordance with the underlying DSP and issues the  
command the underlying voice channel to turn off the echo canceller. When the command has  
been complied with a response is returned from the underlying DSP to the physical interface 708.  
20 The physical interface 708 formats the response into an API message and forwards it to the  
APITask thread 710 which then forwards it as an API response to the API interface 706. The API  
interface 706 correlates the API responses received from the APITask thread 710 to the  
corresponding command/request that prompted the response and forwards a HAPI response to  
the MTA call client 620.

25 Referring to FIG. 27, the described channel associated signaling (CAS) logic 632 utilizes  
a foreign exchange office (FXO) interface to provide exchange of commands and detection of  
events between the MTA call client 620 and standard commercial analog loop/ground start  
devices such as, for example, plain old telephone sets (POTS), group three facsimiles, modems,  
answering machine or other similar devices. The CAS logic 632 can operate in switch mode,  
30 such that ABCD bits 712 are terminated within the CAS logic. The MTA call client interfaces  
with the CAS logic through high level commands 714 and events 716. For example, if the CAS  
logic detects a phone going off the hook, a detection event is forwarded to the MTA call client  
620 which may then notify the call management server as previously described. The MTA call  
client is also able to issue high level commands to the CAS logic.

35 For example, to ring a phone, the MTA call client 620 sends a ring generation profile to  
the CAS logic, and the command is switched into raw CAS bits by FXO termination logic 720  
to control the external interface via a hardware abstraction layer (HAL) 722. Similarly detected  
events, signaled from the hardware SLIC 724, including on/off hook, flash hook and potentially

1 pulse dial. are returned to the FXO termination logic 720 from the hardware abstraction layer as raw CAS bits. The HAL 722 provides a physical interface to the connected SLIC. The HAL 722 formats message signals in accordance with the underlying SLIC and forwards or transmits the message signal to the connected telephony device via the SLIC 724.

5 The FXO termination logic 720 preferably verifies the integrity of the detected event and forwards a detected event message to the MTA call client. The FXO termination logic 720 includes a state machine which monitors detected events signaled from the SLIC 724 via the hardware abstraction layer 722. The FXO termination logic 720 qualifies detected events signaled by the SLIC 724 to determine if a valid on/off hook, flash hook or pulse dial has been detected. 10 The FXO termination logic analyzes the period of the received signal to determine valid flash hook or pulse dial conditions. Similarly, the FXO termination logic 720 includes a state machine which associated with the ringing of connected phones. When the MTA call client issues a command to ring a connected phone with a pre-defined cadence and the associated FXO termination state machine will ring the connected phone in accordance with the MTA command.

15 The described exemplary voice and data processor is preferably implemented with a programmable DSP software architecture (see FIG. 22). The programmable DSP 600 is effectively hidden within the embedded communications software layer. The software layer binds all core DSP algorithms together, interfaces the DSP hardware to the host, and provides low level services such as the allocation of resources to allow higher level software programs to run. 20 An exemplary multi-layer software architecture loaded into the program memory 602 for execution on the DSP platform is shown in FIG.28. The MTA call client 620 provides overall executive control and system management, and directly interfaces a DSP server 730 to the host MIPS core (see to FIG. 3). The DSP server 730 provides DSP resource management and telecommunications signal processing. Operating below the DSP server layer are a number of physical devices (PXD) 732a, 732b, 732c. Each PXD provides an interface between the DSP server 730 and an external telephony device (not shown) via a hardware abstraction layer (HAL) 734. 25

The DSP server 730 includes a resource manager 736 which receives commands from, forwards events to, and exchanges data with the MTA call client 620. The user application layer 30 736 can either be resident on the DSP 600 or alternatively within the MTA call client. An application programming interface 738 (API) provides a software interface between the user MTA call client 620 and the resource manager 736. The resource manager 736 manages the internal / external program and data memory of the DSP 600. In addition the resource manager dynamically allocates DSP resources, performs command routing as well as other general purpose functions. 35

The DSP server 730 also includes virtual device drivers (VHDs) 740a, 740b, 740c. The VHDs are a collection of software objects that control the operation of and provide the facility for real time signal processing. Each VHD 740a, 740b, 740c includes an inbound and outbound

1 media queue (not shown) and a library of signal processing services specific to that VHD 740a,  
740b, 740c. In the described exemplary embodiment, each VHD 740a, 740b, 740c is a complete  
self-contained software module for processing a single channel with a number of different  
telephony devices. Multiple channel capability can be achieved by adding VHDs to the DSP  
5 server 730. The resource manager 736 dynamically controls the creation and deletion of VHDs  
and services.

A switchboard 742 in the DSP server 730 dynamically inter-connects the PXDs 732a,  
732b, 732c with the VHDs 740a, 740b, 740c. Each PXD 732a, 732b, 732c is a collection of  
software objects which provide signal conditioning for one external telephony device. For  
10 example, a PXD may provide volume and gain control for signals from a telephony device prior  
to communication with the switchboard 742. Multiple telephony functionalities can be supported  
on a single channel by connecting multiple PXDs, one for each telephony device, to a single  
VHD via the switchboard 742. Connections within the switchboard 742 are managed by the user  
application layer 738 via a set of API commands to the resource manager 736. The number of  
15 PXDs and VHDs is expandable, and limited only by the memory size and the MIPS (millions  
instructions per second) of the underlying hardware.

A hardware abstraction layer (HAL) 734 interfaces directly with the underlying DSP 600  
hardware (see FIG. 22) and exchanges telephony signals between the external telephony devices  
and the PXDs. The HAL 734 includes basic hardware interface routines, including DSP  
20 initialization, target hardware control, codec sampling, and hardware control interface routines.  
The DSP initialization routine is invoked by the user application layer 738 to initiate the  
initialization of the signal processing system. The DSP initialization sets up the internal registers  
of the signal processing system for memory organization, interrupt handling, timer initialization,  
and DSP configuration. Target hardware initialization involves the initialization of all hardware  
25 devices and circuits external to the signal processing system. The HAL 734 is a physical  
firmware layer that isolates the communications software from the underlying hardware. This  
methodology allows the communications software to be ported to various hardware platforms by  
porting only the affected portions of the HAL 734 to the target hardware.

The exemplary software architecture described above can be integrated into numerous  
30 telecommunications products. In an exemplary embodiment, the software architecture is  
designed to support telephony signals between telephony devices (and/or circuit switched  
networks) and packet based networks. A network VHD (NetVHD) is used to provide a single  
channel of operation and provide the signal processing services for transparently managing voice,

35

1 fax, and modem data across a variety of packet based networks. More particularly, the NetVHD  
encodes and packetizes DTMF, voice, fax, and modem data received from various telephony  
devices and/or circuit switched networks and transmits the packets to the user application layer.  
In addition, the NetVHD disassembles DTMF, voice, fax, and modem data from the user  
5 application layer, decodes the packets into signals, and transmits the signals to the circuit  
switched network or device.

An exemplary embodiment of the NetVHD operating in the described software  
architecture is shown in FIG. 29. The NetVHD includes four operational modes, namely voice  
mode 750, voiceband data mode 752, fax relay mode 754, and data relay mode 756. In each  
10 operational mode, the resource manager invokes various services. For example, in the voice  
mode 750, the resource manager invokes call discrimination 758, packet voice exchange 760,  
and packet tone exchange 762. The packet voice exchange 760 may employ numerous voice  
compression algorithms, including, among others, Linear 128 kbps, G.711 u-law/A-law 64 kbps  
(ITU Recommendation G.711 (1988) - Pulse code modulation (PCM) of voice frequencies),  
15 G.726 16/24/32/40 kbps (ITU Recommendation G.726 (12/90) - 40, 32, 24, 16 kbit/s Adaptive  
Differential Pulse Code Modulation (ADPCM)), G.729A 8 kbps (Annex A (11/96) to ITU  
Recommendation G.729 - Coding of speech at 8 kbit/s using conjugate structure algebraic-code-  
excited linear-prediction (CS-ACELP) - Annex A: Reduced complexity 8 kbit/s CS-ACELP  
speech codec), and G.723 5.3/6.3 kbps (ITU Recommendation G.723.1 (03/96) - Dual rate coder  
20 for multimedia communications transmitting at 5.3 and 6.3 kbit/s). The contents of each of the  
foregoing ITU Recommendations being incorporated herein by reference as if set forth in full.

The packet voice exchange 760 is common to both the voice mode 750 and the voiceband  
data mode 752. In the voiceband data mode 752, the resource manager invokes the packet voice  
exchange 760 for exchanging transparently data without modification (other than packetization)  
25 between the telephony device (or circuit switched network) and the packet based network. This  
is typically used for the exchange of fax and modem data when bandwidth concerns are minimal  
as an alternative to demodulation and remodulation. During the voiceband data mode 752, the  
human speech detector service 768 is also invoked by the resource manager. The human speech  
detector 768 monitors the signal from the near end telephony device for speech. In the event that  
30 speech is detected by the human speech detector 768, an event is forwarded to the resource  
manager which, in turn, causes the resource manager to terminate the human speech detector  
service 768 and invoke the appropriate services for the voice mode 750 (i.e., the call  
discriminator, the packet tone exchange, and the packet voice exchange).

In the fax relay mode 754, the resource manager invokes a packet fax data exchange 764  
35 service. The packet fax exchange 764 may employ various data pumps including, among others,  
V.17 which can operate up to 14,400 bits per second, V.29 which uses a 1700-Hz carrier that is  
varied in both phase and amplitude, resulting in 16 combinations of 8 phases and 4 amplitudes  
which can operate up to 9600 bits per second, and V.27ter which can operate up to 4800 bits per



1 second. Likewise, the resource manager invokes a packet modem data exchange 766 service in  
the data relay mode 756. The packet modem data exchange 766 may employ various data pumps  
including, among others, V.22bis/V.22 with data rates up to 2400 bits per second, V.32bis/V.32  
which enables full-duplex transmission at 14,400 bits per second, and V.34 which operates up  
5 to 33,600 bits per second. The ITU Recommendations setting forth the standards for the  
foregoing data pumps are incorporated herein by reference as if set forth in full.

In the described exemplary embodiment, the user application layer does not need to  
manage any service directly. The user application layer manages the session using high-level  
commands directed to the NetVHD, which in turn directly runs the services. However, the user  
10 application layer can access more detailed parameters of any service if necessary to change, by  
way of example, default functions for any particular application.

In operation, the user application layer opens the NetVHD and connects it to the  
appropriate PXD. The user application then may configure various operational parameters of the  
NetVHD, including, among others, default voice compression (Linear, G.711, G.726, G.723.1,  
15 G.723.1A, G.729A, G.729B), fax data pump (Binary, V.17, V.29, V.27ter), and modem data  
pump (Binary, V.22bis, V.32bis, V.34). The user application layer then loads an appropriate  
signaling service (not shown) into the NetVHD, configures it and sets the NetVHD to the On-  
hook state.

In response to events from the signaling service (not shown) via a near end telephony  
20 device (hookswitch), or signal packets from the far end, the user application will set the NetVHD  
to the appropriate off-hook state, typically voice mode. In an exemplary embodiment, if the  
signaling service event is triggered by the near end telephony device, the packet tone exchange  
will generate dial tone. Once a DTMF tone is detected, the dial tone is terminated. The DTMF  
tones are packetized and forwarded to the user application layer for transmission on the packet  
25 based network. The packet tone exchange could also play ringing tone back to the near end  
telephony device (when a far end telephony device is being rung), and a busy tone if the far end  
telephony device is unavailable. Other tones may also be supported to indicate all circuits are  
busy, or an invalid sequence of DTMF digits were entered on the near end telephony device.

Once a connection is made between the near end and far end telephony devices, the call  
30 discriminator is responsible for differentiating between a voice and machine call by detecting the  
presence of a 2100 Hz. tone (as in the case when the telephony device is a fax or a modem), a  
1100 Hz. tone or V.21 modulated high level data link control (HDLC) flags (as in the case when  
the telephony device is a fax). If a 1100 Hz. tone, or V.21 modulated HDLC flags are detected,  
a calling fax machine is recognized. The NetVHD then terminates the voice mode 36 and  
35 invokes the packet fax exchange to process the call. If however, 2100 Hz tone is detected, the  
NetVHD terminates voice mode and invokes the packet data exchange.

The packet data exchange service further differentiates between a fax and modem by  
continuing to monitor the incoming signal for V.21 modulated HDLC flags, which if present,

1 indicate that a fax connection is in progress. If HDLC flags are detected, the NetVHD terminates packet data exchange service and initiates packet fax exchange service. Otherwise, the packet data exchange service remains operative. In the absence of an 1100 or 2100 Hz. tone, or V.21 modulated HDLC flags the voice mode remains operative.

5 A. The Voice Mode

Voice mode provides signal processing of voice signals. Voice mode enables the transmission of voice over a packet based system such as Voice over IP (VoIP, H.323), Voice over Frame Relay (VoFR, FRF-11), Voice Telephony over ATM (VTOA), or any other proprietary network. The voice mode should also permit voice to be carried over traditional media such as time division multiplex (TDM) networks and voice storage and playback systems.

10 The PXDs for the voice mode provide echo cancellation, gain, and automatic gain control. The network VHD invokes numerous services in the voice mode including call discrimination, packet voice exchange, and packet tone exchange. These network VHD services operate together to provide: (1) an encoder system with DTMF detection, call progress tone detection, voice activity detection, voice compression, and comfort noise estimation, and (2) a decoder system with delay compensation, voice decoding, DTMF generation, comfort noise generation and lost frame recovery.

15 The services invoked by the network VHD in the voice mode and the associated PXD is shown schematically in FIG. 30. In the described exemplary embodiment, the PXD 1060 provides two way communication with a telephone or a circuit switched network, such as a PSTN line (e.g. DS0) carrying a 64kb/s pulse code modulated (PCM) signal, i.e., digital voice samples.

20 The incoming PCM signal 1060a is initially processed by the PXD 1060 to remove far end echos. As the name implies, echos in telephone systems is the return of the talker's voice resulting from the operation of the hybrid with its two-four wire conversion. If there is low end-to-end delay, echo from the far end is equivalent to side-tone (echo from the near-end), and therefore, not a problem. Side-tone gives users feedback as to how loud they are talking, and indeed, without side-tone, users tend to talk too loud. However, far end echo delays of more than about 10 to 30 msec significantly degrade the voice quality and are a major annoyance to the user.

25 An echo canceller 1070 is used to remove echos from far end speech present on the incoming PCM signal 1060a before routing the incoming PCM signal 1060a back to the far end user. The echo canceller 1070 samples an outgoing PCM signal 1060b from the far end user, filters it, and combines it with the incoming PCM signal 1060a. Preferably, the echo canceller 1070 is followed by a non-linear processor (NLP) 1072 which may mute the digital voice samples when far end speech is detected in the absence of near end speech. The echo canceller 1070 may also inject comfort noise which in the absence of near end speech may be roughly at the same level as the true background noise or at a fixed level.

1 After echo cancellation, the power level of the digital voice samples is normalized by an automatic gain control (AGC) 1074 to ensure that the conversation is of an acceptable loudness. Alternatively, the AGC can be performed before the echo canceller 1070, however, this approach would entail a more complex design because the gain would also have to be applied to the  
5 sampled outgoing PCM signal 1060b. In the described exemplary embodiment, the AGC 1074 is designed to adapt slowly, although it should adapt fairly quickly if overflow or clipping is detected. The AGC adaptation should be held fixed if the NLP 1072 is activated.

After AGC, the digital voice samples are placed in the media queue 1066 in the network VHD 1062 via the switchboard 32'. In the voice mode, the network VHD 1062 invokes three  
10 services, namely call discrimination, packet voice exchange, and packet tone exchange. The call discriminator 1068 analyzes the digital voice samples from the media queue to determine whether a 2100 Hz, a 1100 Hz. tone or V.21 modulated HDLC flags are present. As described above with reference to FIG. 4, if either tone or HDLC flags are detected, the voice mode services are terminated and the appropriate service for fax or modem operation is initiated. In the absence  
15 of a 2100 Hz, a 1100 Hz. tone, or HDLC flags, the digital voice samples are coupled to the encoder system which includes a voice encoder 1082, a voice activity detector (VAD) 1080, a comfort noise estimator 1081, a DTMF detector 1076, a call progress tone detector 1077 and a packetization engine 1078.

Typical telephone conversations have as much as sixty percent silence or inactive content. Therefore, high bandwidth gains can be realized if digital voice samples are suppressed during  
20 these periods. A VAD 1080, operating under the packet voice exchange, is used to accomplish this function. The VAD 1080 attempts to detect digital voice samples that do not contain active speech. During periods of inactive speech, the comfort noise estimator 1081 couples silence identifier (SID) packets to a packetization engine 1078. The SID packets contain voice  
25 parameters that allow the reconstruction of the background noise at the far end.

From a system point of view, the VAD 1080 may be sensitive to the change in the NLP 1072. For example, when the NLP 1072 is activated, the VAD 1080 may immediately declare that voice is inactive. In that instance, the VAD 1080 may have problems tracking the true background noise level. If the echo canceller 1070 generates comfort noise during periods of  
30 inactive speech, it may have a different spectral characteristic from the true background noise. The VAD 1080 may detect a change in noise character when the NLP 1072 is activated (or deactivated) and declare the comfort noise as active speech. For these reasons, the VAD 1080 should be disabled when the NLP 1072 is activated. This is accomplished by a "NLP on" message 1072a passed from the NLP 1072 to the VAD 1080.

35 The voice encoder 1082, operating under the packet voice exchange, can be a straight 16 bit PCM encoder or any voice encoder which supports one or more of the standards promulgated by ITU. The encoded digital voice samples are formatted into a voice packet (or packets) by the packetization engine 1078. These voice packets are formatted according to an applications

1 protocol and outputted to the host (not shown). The voice encoder 1082 is invoked only when  
digital voice samples with speech are detected by the VAD 1080. Since the packetization  
interval may be a multiple of an encoding interval, both the VAD 1080 and the packetization  
engine 1078 should cooperate to decide whether or not the voice encoder 1082 is invoked. For  
5 example, if the packetization interval is 10 msec and the encoder interval is 5 msec (a frame of  
digital voice samples is 5 ms), then a frame containing active speech should cause the subsequent  
frame to be placed in the 10 ms packet regardless of the VAD state during that subsequent frame.  
This interaction can be accomplished by the VAD 1080 passing an "active" flag 1080a to the  
packetization engine 1078, and the packetization engine 1078 controlling whether or not the  
10 voice encoder 1082 is invoked.

In the described exemplary embodiment, the VAD 1080 is applied after the AGC 1074.  
This approach provides optimal flexibility because both the VAD 1080 and the voice encoder  
1082 are integrated into some speech compression schemes such as those promulgated in ITU  
Recommendations G.729 with Annex B VAD (March 1996) - Coding of Speech at 8 kbits/s  
15 Using Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP), and G.723.1  
with Annex A VAD (March 1996) - Dual Rate Coder for Multimedia Communications  
Transmitting at 5.3 and 6.3 kbit/s, the contents of which is hereby incorporated by reference as  
through set forth in full herein.

Operating under the packet tone exchange, a DTMF detector 1076 determines whether  
20 or not there is a DTMF signal present at the near end. The DTMF detector 1076 also provides  
a pre-detection flag 1076a which indicates whether or not it is likely that the digital voice sample  
might be a portion of a DTMF signal. If so, the pre-detection flag 1076a is relayed to the  
packetization engine 1078 instructing it to begin holding voice packets. If the DTMF detector  
1076 ultimately detects a DTMF signal, the voice packets are discarded, and the DTMF signal  
25 is coupled to the packetization engine 1078. Otherwise the voice packets are ultimately released  
from the packetization engine 1078 to the host (not shown). The benefit of this method is that  
there is only a temporary impact on voice packet delay when a DTMF signal is pre-detected in  
error, and not a constant buffering delay. Whether voice packets are held while the pre-detection  
flag 1076a is active could be adaptively controlled by the user application layer.

30 Similarly, a call progress tone detector 1077 also operates under the packet tone exchange  
to determine whether a precise signaling tone is present at the near end. Call progress tones are  
those which indicate what is happening to dialed phone calls. Conditions like busy line, ringing  
called party, bad number, and others each have distinctive tone frequencies and cadences  
assigned them. The call progress tone detector 1077 monitors the call progress state, and  
35 forwards a call progress tone signal to the packetization engine to be packetized and transmitted  
across the packet based network. The call progress tone detector may also provide information  
regarding the near end hook status which is relevant to the signal processing tasks. If the hook  
status is on hook, the VAD should preferably mark all frames as inactive, DTMF detection

1 should be disabled, and SID packets should only be transferred if they are required to keep the connection alive.

The decoding system of the network VHD 1062 essentially performs the inverse operation of the encoding system. The decoding system of the network VHD 1062 comprises a  
5 depacketizing engine 1084, a voice queue 1086, a DTMF queue 1088, a precision tone queue 1087, a voice synchronizer 1090, a DTMF synchronizer 1102, a precision tone synchronizer 1103, a voice decoder 1096, a VAD 1098, a comfort noise estimator 1100, a comfort noise generator 1092, a lost packet recovery engine 1094, a tone generator 1104, and a precision tone generator 1105.

10 The depacketizing engine 1084 identifies the type of packets received from the host (i.e., voice packet, DTMF packet, call progress tone packet, SID packet), transforms them into frames which are protocol independent. The depacketizing engine 1084 then transfers the voice frames (or voice parameters in the case of SID packets) into the voice queue 1086, transfers the DTMF frames into the DTMF queue 1088 and transfers the call progress tones into the call progress tone  
15 queue 1087. In this manner, the remaining tasks are, by and large, protocol independent.

A jitter buffer is utilized to compensate for network impairments such as delay jitter caused by packets not arriving at the same time or in the same order in which they were transmitted. In addition, the jitter buffer compensates for lost packets that occur on occasion when the network is heavily congested. In the described exemplary embodiment, the jitter buffer  
20 for voice includes a voice synchronizer 1090 that operates in conjunction with a voice queue 1086 to provide an isochronous stream of voice frames to the voice decoder 1096.

Sequence numbers embedded into the voice packets at the far end can be used to detect lost packets, packets arriving out of order, and short silence periods. The voice synchronizer 1090 can analyze the sequence numbers, enabling the comfort noise generator 1092 during short  
25 silence periods and performing voice frame repeats via the lost packet recovery engine 1094 when voice packets are lost. SID packets can also be used as an indicator of silent periods causing the voice synchronizer 1090 to enable the comfort noise generator 1092. Otherwise, during far end active speech, the voice synchronizer 1090 couples voice frames from the voice queue 1086 in an isochronous stream to the voice decoder 1096. The voice decoder 1096  
30 decodes the voice frames into digital voice samples suitable for transmission on a circuit switched network, such as a 64kb/s PCM signal for a PSTN line. The output of the voice decoder 1096 (or the comfort noise generator 1092 or lost packet recovery engine 1094 if enabled) is written into a media queue 1106 for transmission to the PXD 1060.

35 The comfort noise generator 1092 provides background noise to the near end user during silent periods. If the protocol supports SID packets, (and these are supported for VTOA, FRF-11, and VoIP), the comfort noise estimator at the far end encoding system should transmit SID packets. Then, the background noise can be reconstructed by the near end comfort noise generator 1092 from the voice parameters in the SID packets buffered in the voice queue 1086.

1 However, for some protocols, namely, FRF-11, the SID packets are optional, and other far end  
users may not support SID packets at all. In these systems, the voice synchronizer 1090 must  
continue to operate properly. In the absence of SID packets, the voice parameters of the  
background noise at the far end can be determined by running the VAD 1098 at the voice decoder  
5 1096 in series with a comfort noise estimator 1100.

Preferably, the voice synchronizer 1090 is not dependent upon sequence numbers  
embedded in the voice packet. The voice synchronizer 1090 can invoke a number of mechanisms  
to compensate for delay jitter in these systems. For example, the voice synchronizer 1090 can  
assume that the voice queue 1086 is in an underflow condition due to excess jitter and perform  
10 packet repeats by enabling the lost frame recovery engine 1094. Alternatively, the VAD 1098  
at the voice decoder 1096 can be used to estimate whether or not the underflow of the voice  
queue 1086 was due to the onset of a silence period or due to packet loss. In this instance, the  
spectrum and/or the energy of the digital voice samples can be estimated and the result 1098a fed  
back to the voice synchronizer 1090. The voice synchronizer 1090 can then invoke the lost  
15 packet recovery engine 1094 during voice packet losses and the comfort noise generator 1092  
during silent periods.

When DTMF packets arrive, they are depacketized by the depacketizing engine 1084.  
DTMF frames at the output of the depacketizing engine 1084 are written into the DTMF queue  
1088. The DTMF synchronizer 1102 couples the DTMF frames from the DTMF queue 1088 to  
20 the tone generator 1104. Much like the voice synchronizer, the DTMF synchronizer 1102 is  
employed to provide an isochronous stream of DTMF frames to the tone generator 1104.  
Generally speaking, when DTMF packets are being transferred, voice frames should be  
suppressed. To some extent, this is protocol dependent. However, the capability to flush the  
voice queue 1086 to ensure that the voice frames do not interfere with DTMF generation is  
25 desirable. Essentially, old voice frames which may be queued are discarded when DTMF  
packets arrive. This will ensure that there is a significant inter-digit gap before DTMF tones are  
generated. This is achieved by a "tone present" message 1088a passed between the DTMF queue  
and the voice synchronizer 1090.

The tone generator 1104 converts the DTMF signals into a DTMF tone suitable for a  
30 standard digital or analog telephone. The tone generator 1104 overwrites the media queue 1106  
to prevent leakage through the voice path and to ensure that the DTMF tones are not too noisy.

There is also a possibility that DTMF tone may be fed back as an echo into the DTMF  
detector 1076. To prevent false detection, the DTMF detector 1076 can be disabled entirely (or  
disabled only for the digit being generated) during DTMF tone generation. This is achieved by  
35 a "tone on" message 1104a passed between the tone generator 1104 and the DTMF detector  
1076. Alternatively, the NLP 1072 can be activated while generating DTMF tones.

When call progress tone packets arrive, they are depacketized by the depacketizing engine  
1084. Call progress tone frames at the output of the depacketizing engine 1084 are written into

1 the call progress tone queue 1087. The call progress tone synchronizer 1103 couples the call  
progress tone frames from the call progress tone queue 1087 to a call progress tone generator  
1105. Much like the DTMF synchronizer, the call progress tone synchronizer 1103 is employed  
to provide an isochronous stream of call progress tone frames to the call progress tone generator  
5 1105. And much like the DTMF tone generator, when call progress tone packets are being  
transferred, voice frames should be suppressed. To some extent, this is protocol dependent.  
However, the capability to flush the voice queue 1086 to ensure that the voice frames do not  
interfere with call progress tone generation is desirable. Essentially, old voice frames which may  
be queued are discarded when call progress tone packets arrive to ensure that there is a significant  
10 inter-digit gap before call progress tones are generated. This is achieved by a "tone present"  
message 1087a passed between the call progress tone queue 1087 and the voice synchronizer  
1090.

The call progress tone generator 1105 converts the call progress tone signals into a call  
progress tone suitable for a standard digital or analog telephone. The call progress tone generator  
15 1105 overwrites the media queue 1106 to prevent leakage through the voice path and to ensure  
that the call progress tones are not too noisy.

The outgoing PCM signal in the media queue 1106 is coupled to the PXD 1060 via the  
switchboard 1032'. The outgoing PCM signal is coupled to an amplifier 1108 before being  
outputted on the PCM output line 1060b.

#### 20 1. Echo Canceller with NLP

The problem of line echos such as the reflection of the talker's voice resulting from the  
operation of the hybrid with its two-four wire conversion is a common telephony problem. To  
eliminate or minimize the effect of line echos in the described exemplary embodiment of the  
present invention, an echo canceller with non-linear processing is used. Although echo  
25 cancellation is described in the context of a signal processing system for packet voice exchange,  
those skilled in the art will appreciate that the techniques described for echo cancellation are  
likewise suitable for various applications requiring the cancellation of reflections, or other  
undesirable signals, from a transmission line. Accordingly, the described exemplary embodiment  
for echo cancellation in a signal processing system is by way of example only and not by way of  
30 limitation.

In the described exemplary embodiment the echo canceller preferably complies with one  
or more of the following ITU-T Recommendations G.164 (1988) - Echo Suppressors, G.165  
(March 1993) - Echo Cancellers, and G.168 (April 1997)- Digital Network Echo Cancellers, the  
contents of which are incorporated herein by reference as though set forth in full. The described  
35 embodiment merges echo cancellation and echo suppression methodologies to remove the line  
echos that are prevalent in telecommunication systems. Typically, echo cancellers are favored  
over echo suppressors for superior overall performance in the presence of system noise such as,  
for example, background music, double talk etc., while echo suppressors tend to perform well

1 over a wide range of operating conditions where clutter such as system noise is not present. The  
described exemplary embodiment utilizes an echo suppressor when the energy level of the line  
echo is below the audible threshold, otherwise an echo canceller is preferably used. The use of  
an echo suppressor reduces system complexity, leading to lower overall power consumption or  
5 higher densities (more VHDs per part or network gateway). Those skilled in the art will  
appreciate that various signal characteristics such as energy, average magnitude, echo  
characteristics, as well as information explicitly received in voice or SID packets may be used  
to determine when to bypass echo cancellation. Accordingly, the described exemplary  
embodiment for bypassing echo cancellation in a signal processing system as a function of  
10 estimated echo power is by way of example only and not by way of limitation.

Figure 7 shows the block diagram of an echo canceller in accordance with a preferred  
embodiment of the present invention. If required to support voice transmission via a T1 or other  
similar transmission media, a compressor 1120 may compress the output 1120(a) of the voice  
decoder system into a format suitable for the channel at  $R_{out}$  1120(b). Typically the compressor  
15 1120 provides  $\mu$ -law or A-law compression in accordance with ITU-T standard G.711, although  
linear compression or compression in accordance with alternate companding laws may also be  
supported. The compressed signal at  $R_{out}$  (signal that eventually makes it way to a near end ear  
piece/telephone receiver), may be reflected back as an input signal to the voice encoder system.  
An input signal 1122(a) may also be in the compressed domain (if compressed by compressor  
20 1120) and, if so, an expander 1122 may be required to invert the companding law to obtain a near  
end signal 1122(b). A power estimator 1124 estimates a short term average power 1124(a), a  
long term average power 1124(b), and a maximum power level 1124(c) for the near end signal  
1122(b).

An expander 1126 inverts the companding law used to compress the voice decoder  
25 output signal 1120(b) to obtain a reference signal 1126(a). One of skill in the art will appreciate  
that the voice decoder output signal could alternatively be compressed downstream of the echo  
canceller so that the expander 1126 would not be required. However, to ensure that all non-  
linearities in the echo path are accounted for in the reference signal 1126(a) it is preferable to  
compress / expand the voice decoder output signal 1120(b). A power estimator 1128 estimates  
30 a short term average power 1128(a), a long term average power 1128(b), a maximum power  
level 1128(c) and a background power level 1128(d) for the reference signal 1126(a). The  
reference signal 1126(a) is input into a finite impulse response (FIR) filter 1130. The FIR filter  
1130 models the transfer characteristics of a dialed telephone line circuit so that the unwanted  
echo may preferably be canceled by subtracting filtered reference signal 1130(a) from the near  
35 end signal 1122(b) in a difference operator 1132.

However, for a variety of reasons, such as for example, non-linearities in the hybrid and  
tail circuit, estimation errors, noise in the system, etc., the adaptive FIR filter 1130 may not  
identically model the transfer characteristics of the telephone line circuit so that the echo



1 canceller may be unable to cancel all of the resulting echo. Therefore, a non linear processor (NLP) 1140 is used to suppress the residual echo during periods of far end active speech with no  
 near end speech. During periods of inactive speech, a power estimator 1138 estimates the  
 performance of the echo canceller by estimating a short term average power 1138(a), a long term  
 5 average power 1138(b) and background power level 1138(c) for an error signal 1132(b) which  
 is an output of the difference operator 1132. The estimated performance of the echo canceller is  
 one measure utilized by adaptation logic 1136 to selectively enable a filter adapter 1134 which  
 controls the convergence of the adaptive FIR filter 1130. The adaptation logic 1136 processes  
 the estimated power levels of the reference signal (1128a, 1128b, 1128c and 1128d) the near end  
 10 signal (1124a, 1124b and 1124c) and the error signal (1138a, 1138b and 1138c) to control the  
 invocation of the filter adapter 1134 as well as the step size to be used during adaptation.

In the described preferred embodiment, the echo suppressor is a simple bypass 1144(a)  
 that is selectively enabled by toggling the bypass cancellation switch 1144. A bypass estimator  
 1142 toggles the bypass cancellation switch 1144 based upon the maximum power level 1128(c)  
 15 of the reference signal 1126(a), the long term average power 1138(b) of the error signal 1132(b)  
 and the long term average power 1124(b) of the near end signal 1122(b). One skilled in the art  
 will appreciate that a NLP or other suppressor could be included in the bypass path 1144(a), so  
 that the described echo suppressor is by way of example only and not by way of limitation.

In an exemplary embodiment, the adaptive filter 1130 models the transfer characteristics  
 20 of the hybrid and the tail circuit of the telephone circuit. The tail length supported should  
 preferably be at least 16 msec. The adaptive filter 1130 may be a linear transversal filter or other  
 suitable finite impulse response filter. In the described exemplary embodiment, the echo  
 canceller preferably converges or adapts only in the absence of near end speech. Therefore, near  
 end speech and/or noise present on the input signal 1122(a) may cause the filter adapter 1134 to  
 25 diverge. To avoid divergence the filter adapter 1134 is preferably selectively enabled by the  
 adaptation logic 1136. In addition, the time required for an adaptive filter to converge increases  
 significantly with the number of coefficients to be determined. Reasonable modeling of the  
 hybrid and tail circuits with a finite impulse response filter requires a large number of coefficients  
 so that filter adaptation is typically computationally intense. In the described exemplary  
 30 embodiment the DSP resources required for filter adaptation are minimized by adjusting the  
 adaptation speed of the FIR filter 1130.

The filter adapter 1134 is preferably based upon a normalized least mean square algorithm  
 (NLMS) as described in S. Haykin, Adaptive Filter Theory, and T. Parsons, Voice and Speech  
 Processing, the contents of which are incorporated herein by reference as if set forth in full. The  
 35 error signal 1132(b) at the output of the difference operator 1132 for the adaptation logic may  
 preferably be characterized as follows:

$$e(n) = s(n) - \sum c(j)r(n-j)$$

where  $e(n)$  is the error signal at time  $n$ ,  $r(n)$  is the reference signal 1126(a) at time  $n$  and

1  $s(n)$  is the near end signal 1122(b) at time  $n$ , and  $c(j)$  are the coefficients of the transversal filter where the dimension of the transversal filter is preferably the worst case echo path length (i.e. the length of the tail circuit  $L$ ) and  $c(j)$ , for  $j=0$  to  $L-1$ . is given by:

$$c(j) = c(j) + \mu * e(n) * r(n - j)$$

5 wherein  $c(j)$  is preferably initialized to a reasonable value such as for example zero.

Assuming a block size of one msec (or 8 samples at a sampling rate of 8 kHz), the short term average power of the reference signal  $P_{ref}$  is the sum of the last  $L$  reference samples and the energy for the current eight samples so that

$$\mu = \frac{P_{ref}(n)}{\alpha}$$

10 where  $\alpha$  is the adaptation step size. One of skill in the art will appreciate that the filter adaptation logic may be implemented in a variety of ways, including fixed point rather than the described floating point realization. Accordingly, the described exemplary adaptation logic is by way of example only and not by way of limitation.

To support filter adaptation the described exemplary embodiment includes the power estimator 1128 that estimates the short term average power 1128(a) of the reference signal 1126(a) ( $P_{ref}$ ). In the described exemplary embodiment the short term average power is preferably estimated over the worst case length of the echo path plus eight samples, (i.e. the length of the FIR filter  $L + 8$  samples). In addition, the power estimator 1128 computes the maximum power level 1128(c) of the reference signal 1126(a) ( $P_{refmax}$ ) over a period of time that is preferably equal to the tail length  $L$  of the echo path. For example, putting a time index on the short term average power, so that  $P_{ref}(n)$  is the power of the reference signal at time  $n$ .  $P_{refmax}$  is then characterized as:

$$P_{refmax}(n) = \max P_{ref}(j) \text{ for } j = n - L_{msec} \text{ to } j = n$$

25 where  $L_{msec}$  is the length of the tail in msec so that  $P_{refmax}$  is the maximum power in the reference signal  $P_{ref}$  over a length of time equal to the tail length.

The second power estimator 1124 estimates the short term average power of the near end signal 1122(b) ( $P_{near}$ ) in a similar manner. The short term average power 1128(a) of the error signal 1132(b) (the output of difference operator 1132),  $P_{err}$  is also estimated in a similar manner by the third power estimator 1138.

30 In addition, the echo return loss (ERL), defined as the loss from  $R_{out}$  1120(b) to  $S_{in}$  1122(a) in the absence of near end speech, is periodically estimated and updated. In the described exemplary embodiment the ERL is estimated and updated about every 5-20 msec. The power estimator 1128 estimates the long term average power 1128(b) ( $P_{refERL}$ ) of the reference signal 1126(a) in the absence of near end speech. The second power estimator 1124 estimates the long term average power 1124(b) ( $P_{nearERL}$ ) of the near end signal 1122(b) in the absence of near end speech. The adaptation logic 1136 computes the ERL by dividing the long term average power of the reference signal ( $P_{refERL}$ ) by the long term average power of the near end signal ( $P_{nearERL}$ ). The adaptation logic 1136 preferably only updates the long term averages used

1 to compute the estimated ERL if the estimated short term power level 1128(a) ( $P_{ref}$ ) of the  
reference signal 1126(a) is greater than a predetermined threshold, preferably in the range of  
about -30 to -35 dBm0; and the estimated short term power level 1128(a) ( $P_{ref}$ ) of the reference  
signal 1126(a) is preferably larger than about at least the short term average power 1124(a) ( $P_{near}$ )  
5 of the near end signal 1122(b) ( $P_{ref} > P_{near}$  in the preferred embodiment).

In the preferred embodiment, the long term averages ( $P_{refERL}$  and  $P_{nearERL}$ ) are based on a  
first order infinite impulse response (IIR) recursive filter, wherein the inputs to the two first order  
filters are  $P_{ref}$  and  $P_{near}$ .

$$P_{nearERL} = (1-\beta) * P_{nearERL} + P_{near} * \beta; \text{ and}$$

$$P_{refERL} = (1-\beta) * P_{refERL} + P_{ref} * \beta$$

where filter coefficient  $\beta = 1/64$

Similarly, the adaptation logic 1136 of the described exemplary embodiment characterizes  
the effectiveness of the echo canceller by estimating the echo return loss enhancement (ERLE).  
The ERLE is an estimation of the reduction in power of the near end signal 1122(b) due to echo  
15 cancellation when there is no near end speech present. The ERLE is the average loss from the  
input 1132(a) of the difference operator 1132 to the output 1132(b) of the difference operator  
1132. The adaptation logic 1136 in the described exemplary embodiment periodically estimates  
and updates the ERLE, preferably in the range of about 5 to 20 msec. In operation, the power  
estimator 1124 estimates the long term average power 1124(b)  $P_{nearERLE}$  of the near end signal  
20 1122(b) in the absence of near end speech. The power estimator 1138 estimates the long term  
average power 1138(b)  $P_{errERLE}$  of the error signal 1132(b) in the absence of near end speech. The  
adaptation logic 1136 computes the ERLE by dividing the long term average power 1124(a)  
 $P_{nearERLE}$  of the near end signal 1122(b) by the long term average power 1138(b)  $P_{errERLE}$  of the  
error signal 1132(b). The adaptation logic 1136 preferably updates the long term averages used  
25 to compute the estimated ERLE only when the estimated short term average power 1128(a) ( $P_{ref}$ )  
of the reference signal 1126(a) is greater than a predetermined threshold preferably in the range  
of about -30 to -35 dBm0; and the estimated short term average power 1124(a) ( $P_{near}$ ) of the near  
end signal 1122(b) is large as compared to the estimated short term average power 1138(a) ( $P_{err}$ )  
of the error signal (preferably when  $P_{near}$  is approximately greater than or equal to four times the  
30 short term average power of the error signal ( $4P_{err}$ )). Therefore, an ERLE of approximately 6  
dB is preferably required before the ERLE tracker will begin to function.

In the preferred embodiment, the long term averages ( $P_{nearERLE}$  and  $P_{errERLE}$ ) may be based  
on a first order IIR (infinite impulse response) recursive filter, wherein the inputs to the two first  
order filters are  $P_{near}$  and  $P_{err}$ .

$$P_{nearERLE} = (1-\beta) * P_{nearERL} + P_{near} * \beta; \text{ and}$$

$$P_{errERLE} = (1-\beta) * P_{errERL} + P_{err} * \beta$$

where filter coefficient  $\beta = 1/64$

It should be noted that  $P_{nearERL} \neq P_{nearERLE}$  because the conditions under which each

1 is updated are different.

To assist in the determination of whether to invoke the echo canceller and if so with what step size, the described exemplary embodiment estimates the power level of the background noise. The power estimator 1128 tracks the long term energy level of the background noise 1128(d) ( $B_{ref}$ ) of the reference signal 1126(a). The power estimator 1128 utilizes a much faster time constant when the input energy is lower than the background noise estimate (current output). With a fast time constant the power estimator 1128 tends to track the minimum energy level of the reference signal 1126(a). By definition, this minimum energy level is the energy level of the background noise of the reference signal  $B_{ref}$ . The energy level of the background noise of the error signal  $B_{err}$  is calculated in a similar manner. The estimated energy level of the background noise of the error signal ( $B_{err}$ ) is not updated when the energy level of the reference signal is larger than a predetermined threshold (preferably in the range of about 30-35 dBm0).

In addition, the invocation of the echo canceller depends on whether near end speech is active. Preferably, the adaptation logic 1136 declares near end speech active when three conditions are met. First, the short term average power of the error signal should preferably exceed a minimum threshold, preferably on the order of about -36 dBm0 ( $P_{err} \geq -36$  dBm0). Second, the short term average power of the error signal should preferably exceed the estimated power level of the background noise for the error signal by preferably at least about 6 dB ( $P_{err} \geq B_{err} + 6$  dB). Third, the short term average power 1124(a) of the near end signal 1122(b) is preferably approximately 3 dB greater than the maximum power level 1128(c) of the reference signal 1126(a) less the estimated ERL ( $P_{near} \geq P_{refmax} - ERL + 3$  dB). The adaptation logic 1136 preferably sets a near end speech hangover counter (not shown) when near end speech is detected. The hangover counter is used to prevent clipping of near end speech by delaying the invocation of the NLP 1140 when near end speech is detected. Preferably the hangover counter is on the order of about 150 msec.

In the described exemplary embodiment, if the maximum power level ( $P_{refmax}$ ) of the reference signal minus the estimated ERL is less than the threshold of hearing (all in dB) neither echo cancellation or non-linear processing are invoked. In this instance, the energy level of the echo is below the threshold of hearing, typically about -65 to -69 dBm0, so that echo cancellation and non-linear processing are not required for the current time period. Therefore, the bypass estimator 1142 sets the bypass cancellation switch 1144 in the down position, so as to bypass the echo canceller and the NLP and no processing (other than updating the power estimates) is performed. Also, if the maximum power level ( $P_{refmax}$ ) of the reference signal minus the estimated ERL is less than the maximum of either the threshold of hearing, or background power level  $B_{err}$  of the error signal minus a predetermined threshold ( $P_{refmax} - ERL < \text{threshold of hearing or } (B_{err} - \text{threshold})$ ) neither echo cancellation or non-linear processing are invoked. In this instance, the echo is buried in the background noise or below the threshold of hearing, so that echo cancellation and non-linear processing are not required for the current time period. In the

described preferred embodiment the background noise estimate is preferably greater than the threshold of hearing, such that this is a broader method for setting the bypass cancellation switch. The threshold is preferably in the range of about 8-12 dB.

Similarly, if the maximum power level ( $P_{\text{refmax}}$ ) of the reference signal minus the estimated ERL is less than the short term average power  $P_{\text{near}}$  minus a predetermined threshold ( $P_{\text{refmax}} - \text{ERL} < P_{\text{near}} - \text{threshold}$ ) neither echo cancellation or non-linear processing are invoked. In this instance, it is highly probable that near end speech is present, and that such speech will likely mask the echo. This method operates in conjunction with the above described techniques for bypassing the echo canceller and NLP. The threshold is preferably in the range of about 8-12 dB. If the NLP contains a real comfort noise generator, i.e., a non-linearity which mutes the incoming signal and injects comfort noise of the appropriate character then a determination that the NLP will be invoked in the absence of filter adaptation allows the adaptive filter to be bypassed or not invoked. This method is, used in conjunction with the above methods. If the adaptive filter is not executed then adaptation does not take place, so this method is preferably used only when the echo canceller has converged.

If the bypass cancellation switch 1144 is in the down position, the adaptation logic 1136 disables the filter adapter 1134. Otherwise, for those conditions where the bypass cancellation switch 1144 is in the up position so that both adaptation and cancellation may take place, the operation of the preferred adaptation logic 1136 proceeds as follows:

If the estimated echo return loss enhancement is low (preferably in the range of about 0-9dBm) the adaptation logic 1136 enables rapid convergence with an adaptation step size  $\alpha = 1/4$ . In this instance, the echo canceller is not converged so that rapid adaptation is warranted. However, if near end speech is detected within the hangover period, the adaptation logic 1136 either disables adaptation or uses very slow adaptation, preferably an adaptation speed on the order of about one-eighth that used for rapid convergence or an adaptation step size  $\alpha = 1/32$ . In this case the adaptation logic 1136 disables adaptation when the echo canceller is converged. Convergence may be assumed if adaptation has been active for a total of one second after the off hook transition or subsequent to the invocation of the echo canceller. Otherwise if the combined loss (ERL+ERLE) is in the range of about 33-36 dB, the adaptation logic 1136 enables slow adaptation (preferably one-eighth the adaptation speed of rapid convergence or an adaptation step size  $\alpha = 1/32$ ). If the combined loss (ERL+ERLE) is in the range of about 23-33 dB, the adaptation logic 1136 enables a moderate convergence speed, preferably on the order of about one-fourth the adaptation speed used for rapid convergence or an adaptation step size  $\alpha = 1/16$ .

Otherwise, one of three preferred adaptation speeds is chosen based on the estimated echo power ( $P_{\text{refmax}}$  minus the ERL) in relation to the power level of the background noise of the error signal. If the estimated echo power ( $P_{\text{refmax}} - \text{ERL}$ ) is large compared to the power level of the background noise of the error signal ( $P_{\text{refmax}} - \text{ERL} \geq B_{\text{err}} + 24 \text{ dB}$ ), rapid adaptation / convergence is enabled with an adaptation step size on the order of about  $\alpha = 1/4$ . Otherwise, if ( $P_{\text{refmax}} - \text{ERL}$

1  $\geq B_{\text{err}} + 18 \text{ dB}$ ) the adaptation speed is reduced to approximately one-half the adaptation speed used for rapid convergence or an adaptation step size on the order of about  $\alpha = 1/8$ . Otherwise, if  $(P_{\text{refmax}} - \text{ERL} \geq B_{\text{err}} + 9 \text{ dB})$  the adaptation speed is further reduced to approximately one-quarter the adaptation speed used for rapid convergence or an adaptation step size  $\alpha = 1/16$ .

5 As a further limit on adaptation speed, if echo canceller adaptation has been active for a sum total of one second since initialization or an off-hook condition then the maximum adaptation speed is limited to one-fourth the adaptation speed used for rapid convergence ( $\alpha = 1/16$ ). Also, if the echo path changes appreciably or if for any reason the estimated ERLE is negative, (which typically occurs when the echo path changes) then the coefficients are cleared and an adaptation counter is set to zero (the adaptation counter measures the sum total of  
10 adaptation cycles in samples).

The NLP 1140 is a two state device. The NLP 1140 is either on (applying non-linear processing) or it is off (applying unity gain). When the NLP 1140 is on it tends to stay on, and when the NLP 1140 is off it tends to stay off. The NLP 1140 is preferably invoked when the  
15 bypass cancellation switch 1144 is in the upper position so that adaptation and cancellation are active. Otherwise, the NLP 1140 is not invoked and the NLP 1140 is forced into the off state.

Initially, a stateless first NLP decision is created. The decision logic is based on three decision variables (D1-D3). The decision variable D1 is set if it is likely that the far end is active (i.e. the short term average power 1128(a) of the reference signal 1126(a) is preferably about 6  
20 dB greater than the power level of the background noise 1128(d) of the reference signal), and the short term average power 1128(a) of the reference signal 1126(a) minus the estimated ERL is greater than the estimated short term average power 1124(a) of the near end signal 1122(b) minus a small threshold, preferably in the range of about 6 dB. In the preferred embodiment, this is represented by:  $(P_{\text{ref}} \geq B_{\text{ref}} + 6 \text{ dB})$  and  $((P_{\text{ref}} - \text{ERL}) \geq (P_{\text{near}} - 6 \text{ dB}))$ . Thus, decision variable D1 attempts to detect far end active speech and high ERL (implying no near end). Preferably,  
25 decision variable D2 is set if the power level of the error signal is on the order of about 9 dB below the power level of the estimated short term average power 1124(a) of the near end signal 1122(b) (a condition that is indicative of good short term ERLE). In the preferred embodiment,  $P_{\text{err}} \leq P_{\text{near}} - 9 \text{ dB}$  is used (a short term ERLE of 9 dB). The third decision variable D3 is preferably set if the combined loss (reference power to error power) is greater than a threshold. In the preferred embodiment, this is:  $P_{\text{err}} \leq P_{\text{ref}} - t$ , where  $t$  is preferably initialized to about 6 dB and preferably increases to about 12 dB after about one second of adaptation. (In other words,  
30 it is only adapted while convergence is enabled).

The third decision variable D3 results in more aggressive non linear processing while the  
35 echo canceller is uncovered. Once the echo canceller converges, the NLP 1140 can be slightly less aggressive. The initial stateless decision is set if two of the sub-decisions or control variables are initially set. The initial decision set implies that the NLP 1140 is in a transition state or remaining on.

1 A NLP state machine (not shown) controls the invocation and termination of NLP 1140  
in accordance with the detection of near end speech as previously described. The NLP state  
machine delays activation of the NLP 1140 when near end speech is detected to prevent clipping  
the near end speech. In addition, the NLP state machine is sensitive to the near end speech  
5 hangover counter (set by the adaptation logic when near end speech is detected) so that activation  
of the NLP 1140 is further delayed until the near end speech hangover counter is cleared. The  
NLP state machine also deactivates the NLP 1140. The NLP state machine preferably sets an off  
counter when the NLP 1140 has been active for a predetermined period of time, preferably about  
the tail length in msec. The "off" counter is cleared when near end speech is detected and  
10 decremented while non-zero when the NLP is on. The off counter delays termination of NLP  
processing when the far end power decreases so as to prevent the reflection of echo stored in the  
tail circuit. If the near end speech detector hangover counter is on, the above NLP decision is  
overridden and the NLP is forced into the off state.

In the preferred embodiment, the NLP 1140 may be implemented with a suppressor that  
15 adaptively suppresses down to the background noise level ( $B_{err}$ ), or a suppressor that suppresses  
completely and inserts comfort noise with a spectrum that models the true background noise.

## 2. Automatic Gain Control

In an exemplary embodiment of the present invention, AGC is used to normalize digital  
voice samples to ensure that the conversation between the near and far end users is maintained  
20 at an acceptable volume. The described exemplary embodiment of the AGC includes a signal  
bypass for the digital voice samples when the gain adjusted digital samples exceeds a  
predetermined power level. This approach provides rapid response time to increased power  
levels by coupling the digital voice samples directly to the output of the AGC until the gain falls  
off due to AGC adaptation. Although AGC is described in the context of a signal processing  
25 system for packet voice exchange, those skilled in the art will appreciate that the techniques  
described for AGC are likewise suitable for various applications requiring a signal bypass when  
the processing of the signal produces undesirable results. Accordingly, the described exemplary  
embodiment for AGC in a signal processing system is by way of example only and not by way  
of limitation.

30 In an exemplary embodiment, the AGC can be either fully adaptive or have a fixed gain.  
Preferably, the AGC supports a fully adaptive operating mode with a range of about -30 dB to  
30 dB. A default gain value may be independently established, and is typically 0 dB. If adaptive  
gain control is used, the initial gain value is specified by this default gain. The AGC adjusts the  
gain factor in accordance with the power level of an input signal. Input signals with a low energy  
35 level are amplified to a comfortable sound level, while high energy signals are attenuated.

A block diagram of a preferred embodiment of the AGC is shown in FIG. 32. A  
multiplier 1150 applies a gain factor 1152 to an input signal 1150(a) which is then output to the  
media queue 1066 of the network VHD (see FIG. 30). The default gain, typically 0 dB is initially

1 applied to the input signal 1150(a). A power estimator 1154 estimates the short term average  
power 1154(a) of the gain adjusted signal 1150(b). The short term average power of the input  
signal 1150(a) is preferably calculated every eight samples, typically every one ms for a 8 kHz  
signal. Clipping logic 1156 analyzes the short term average power 1154(a) to identify gain  
5 adjusted signals 1150(b) whose amplitudes are greater than a predetermined clipping threshold.  
The clipping logic 1156 controls an AGC bypass switch 1157, which directly connects the input  
signal 1150(a) to the media queue 1066 when the amplitude of the gain adjusted signal 150(b)  
exceeds the predetermined clipping threshold. The AGC bypass switch 1157 remains in the up  
or bypass position until the AGC adapts so that the amplitude of the gain adjusted signal 1150(b)  
10 falls below the clipping threshold.

The power estimator 1154 also calculates a long term average power 1154(b) for the input  
signal 1150(a), by averaging thirty two short term average power estimates, (i.e. averages thirty  
two blocks of eight samples). The long term average power is a moving average which provides  
significant hangover. A peak tracker 1158 utilizes the long term average power 1154(b) to  
15 calculate a reference value which gain calculator 1160 utilizes to estimate the required adjustment  
to a gain factor 1152. The gain factor 1152 is applied to the input signal 1150(a) by the  
multiplier 1150. In the described exemplary embodiment the peak tracker 1158 may preferably  
be a non-linear filter. The peak tracker 1158 preferably stores a reference value which is  
dependent upon the last maximum peak. The peak tracker 1158 compares the long term average  
20 power estimate to the reference value. FIG. 33 shows the peak tracker output as a function of an  
input signal, demonstrating that the reference value that the peak tracker 1158 forwards to the  
gain calculator 1160 should preferably rise quickly if the signal amplitude increases, but  
decrement slowly if the signal amplitude decreases. Thus for active voice segments followed by  
silence, the peak tracker output slowly decreases, so that the gain factor applied to the input  
25 signal 1150(a) may be slowly increased. However, for long inactive or silent segments followed  
by loud or high amplitude voice segments, the peak tracker output increases rapidly, so that the  
gain factor applied to the input signal 1150(a) may be quickly decreased.

In the described exemplary embodiment, the peak tracker should be updated when the  
estimated long term power exceeds the threshold of hearing. Peak tracker inputs include the  
30 current estimated long term power level  $a(i)$ , the previous long term power estimate,  $a(i-1)$ , and  
the previous peak tracker output  $x(i-1)$ . In operation, when the long term energy is varying  
rapidly, preferably when the previous long term power estimate is on the order of four times  
greater than the current long term estimate or vice versa, the peak tracker should go into hangover  
mode. In hangover mode, the peak tracker should not be updated. The hangover mode prevents  
35 adaptation on impulse noise.

If the long term energy estimate is large compared to the previous peak tracker estimate,  
then the peak tracker should adapt rapidly. In this case the current peak tracker output  $x(i)$  is  
given by:



1            $x(i) = (7x(i-1) + a(i))/8.$

          where  $x(i-1)$  is the previous peak tracker output and  $a(i)$  is the current long term power estimate.

          If the long term energy is less than the previous peak tracker output, then the peak tracker will adapt slowly. In this case the current peak tracker output  $x(i)$  is given by:

5            $x(i) = x(i-1) * 255/256.$

          Referring to FIG. 34, a preferred embodiment of the gain calculator 1160 slowly increments the gain factor 1152 for signals below the comfort level of hearing 1166 (below minVoice) and decrements the gain for signals above the comfort level of hearing 1164 (above MaxVoice). The described exemplary embodiment of the gain calculator 1160 decrements the gain factor 1152 for signals above the clipping threshold relatively fast, preferably on the order of about 2-4 dB/sec, until the signal has been attenuated approximately 10 dB or the power level of the signal drops to the comfort zone. The gain calculator 1160 preferably decrements the gain factor 1152 for signals with power levels that are above the comfort level of hearing 1164 (MaxVoice) but below the clipping threshold 1166 (Clip) relatively slowly, preferably on the order of about 0.1-0.3 dB/sec until the signal has been attenuated approximately 4 dB or the power level of the signal drops to the comfort zone.

          The gain calculator 1160 preferably does not adjust the gain factor 1152 for signals with power levels within the comfort zone (between minVoice and MaxVoice), or below the maximum noise power threshold 1168 (MaxNoise). The preferred values of MaxNoise, minVoice, MaxVoice, Clip are related to a noise floor 1170 and are preferably in 3dB increments. The noise floor is preferably empirically derived by calibrating the host DSP platform with a known load. The noise floor preferably adjustable and is typically within the range of about, -45 to -52 dBm. A MaxNoise value of two corresponds to a power level 6 dB above the noise floor 1170, whereas a clip level of nine corresponds to 27 dB above noise floor 1170. For signals with power levels below the comfort zone (less than minVoice) but above the maximum noise threshold, the gain calculator 1160 preferably increments the gain factor 1152 logarithmically at a rate of about 0.1-0.3 dB/sec, until the power level of the signal is within the comfort zone or a gain of approximately 10 dB is reached.

          In the described exemplary embodiment, the AGC is designed to adapt slowly, although it should adapt fairly quickly if overflow or clipping is detected. From a system point of view, AGC adaptation should be held fixed if the NLP 1072 (see FIG. 30) is activated or the VAD 1080 (see FIG. 30) determines that voice is inactive. In addition, the AGC is preferably sensitive to the amplitude of received call progress tones. In the described exemplary embodiment, rapid adaptation may be enabled as a function of the actual power level of a received call progress tone such as for example a ring back tone, compared to the power levels set forth in the applicable standards.

### 3.     Voice Activity Detector

1 In an exemplary embodiment, the VAD, in either the encoder system or the decoder  
system, can be configured to operate in multiple modes so as to provide system tradeoffs between  
voice quality and bandwidth requirements. In a first mode, the VAD is always disabled and  
declares all digital voice samples as active speech. This mode is applicable if the signal  
5 processing system is used over a TDM network, a network which is not congested with traffic,  
or when used with PCM (ITU Recommendation G.711 (1988) - Pulse Code Modulation (PCM)  
of Voice Frequencies, the contents of which is incorporated herein by reference as if set forth in  
full) in a PCM bypass mode for supporting data or fax modems.

10 In a second "transparent" mode, the voice quality is indistinguishable from the first mode.  
In transparent mode, the VAD identifies digital voice samples with an energy below the threshold  
of hearing as inactive speech. The threshold may be adjustable between -90 and -40 dBm with  
a default value of -60 dBm. The transparent mode may be used if voice quality is much more  
important than bandwidth. This may be the case, for example, if a G.711 voice encoder (or  
decoder) is used.

15 In a third "conservative" mode, the VAD identifies low level (but audible) digital voice  
samples as inactive, but will be fairly conservative about discarding the digital voice samples.  
A low percentage of active speech will be clipped at the expense of slightly higher transmit  
bandwidth. In the conservative mode, a skilled listener may be able to determine that voice  
activity detection and comfort noise generation is being employed. The threshold for the  
20 conservative mode may preferably be adjustable between -65 and -35 dBm with a default value  
of -60 dBm.

In a fourth "aggressive" mode, bandwidth is at a premium. The VAD is aggressive about  
discarding digital voice samples which are declared inactive. This approach will result in speech  
being occasionally clipped, but system bandwidth will be vastly improved. The threshold for the  
25 aggressive mode may preferably be adjustable between -60 and -30 dBm with a default value  
of -55 dBm.

30 The transparent mode is typically the default mode when the system is operating with 16  
bit PCM, companded PCM (G.711) or adaptive differential PCM (ITU Recommendations G.726  
(Dec. 1990) - 40, 32, 24, 16 kbit/s Using Low-Delay Code Excited Linear Prediction, and G.727  
(Dec. 1990) - 5 -, 4 -, 3 -, and 2 - Sample Embedded Adaptive Differential Pulse Code  
Modulation). In these instances, the user is most likely concerned with high quality voice since  
a high bit-rate voice encoder (or decoder) has been selected. As such, a high quality VAD should  
be employed. The transparent mode should also be used for the VAD operating in the decoder  
system since bandwidth is not a concern (the VAD in the decoder system is used only to update  
35 the comfort noise parameters). The conservative mode could be used with ITU Recommendation  
G.728 (Sept. 1992) - Coding of Speech at 16 kbit/s Using Low-Delay Code Excited Linear  
Prediction, G.729, and G.723.1. For systems demanding high bandwidth efficiency, the  
aggressive mode can be employed as the default mode.

1       The mechanism in which the VAD detects digital voice samples that do not contain active  
speech can be implemented in a variety of ways. One such mechanism entails monitoring the  
energy level of the digital voice samples over short periods (where a period length is typically  
in the range of about 10 to 30 msec). If the energy level exceeds a fixed threshold, the digital  
5       voice samples are declared active, otherwise they are declared inactive. The transparent mode  
can be obtained when the threshold is set to the threshold level of hearing.

      Alternatively, the threshold level of the VAD can be adaptive and the background noise  
energy can be tracked. If the energy in the current period is sufficiently larger than the  
background noise estimate by the comfort noise estimator, the digital voice samples are declared  
10      active, otherwise they are declared inactive. The VAD may also freeze the comfort noise  
estimator or extend the range of active periods (hangover). This type of VAD is used in GSM  
(European Digital Cellular Telecommunications System; Half rate Speech Part 6: Voice Activity  
Detector (VAD) for Half Rate Speech Traffic Channels (GSM 6.42), the contents of which is  
incorporated herein by reference as if set forth in full) and QCELP (W. Gardner, P. Jacobs, and  
15      C. Lee, "QCELP: A Variable Rate Speech Coder for CDMA Digital Cellular," in *Speech and  
Audio Coding for Wireless and Network Applications*, B.S. atal, V. Cuperman, and A. Gersho  
(eds.), the contents of which is incorporated herein by reference as if set forth in full).

      In a VAD utilizing an adaptive threshold level, speech parameters such as the zero  
crossing rate, spectral tilt, energy and spectral dynamics are measured and compared to stored  
20      values for noise. If the parameters differ significantly from the stored values, it is an indication  
that active speech is present even if the energy level of the digital voice samples is low.

      When the VAD operates in the conservative or transparent mode, measuring the energy  
of the digital voice samples can be sufficient for detecting inactive speech. However, the spectral  
dynamics of the digital voice samples against a fixed threshold may be useful in discriminating  
25      between long voice segments with audio spectra and long term background noise. In an  
exemplary embodiment of a VAD employing spectral analysis, the VAD performs auto-  
correlations using Itakura or Itakura-Saito distortion to compare long term estimates based on  
background noise to short term estimates based on a period of digital voice samples. In addition,  
if supported by the voice encoder, line spectrum pairs (LSPs) can be used to compare long term  
30      LSP estimates based on background noise to short terms estimates based on a period of digital  
voice samples. Alternatively, FFT methods can be are used when the spectrum is available from  
another software module.

      Preferably, hangover should be applied to the end of active periods of the digital voice  
samples with active speech. Hangover bridges short inactive segments to ensure that quiet  
35      trailing, unvoiced sounds (such as /s/), are classified as active. The amount of hangover can be  
adjusted according to the mode of operation of the VAD. If a period following a long active  
period is clearly inactive (i.e., very low energy with a spectrum similar to the measured  
background noise) the length of the hangover period can be reduced. Generally, a range of about

1 40 to 300 msec of inactive speech following an active speech burst will be declared active speech due to hangover.

#### 4. Comfort Noise Generator

5 According to industry research the average voice conversation includes as much as sixty percent silence or inactive content so that transmission across the packet based network can be significantly reduced if non-active speech packets are not transmitted across the packet based network. In an exemplary embodiment of the present invention, a comfort noise generator is used to effectively reproduce background noise when non-active speech packets are not received. In the described preferred embodiment, comfort noise is generated as a function signal characteristics received from a remote source and estimated signal characteristics. In the  
10 described exemplary embodiment comfort noise parameters are preferably generated by a comfort noise estimator. The comfort noise parameters may be transmitted from the far end or can be generated by monitoring the energy level and spectral characteristics of the far end noise at the end of active speech (i.e., during the hangover period). Although comfort noise generation is described in the context of a signal processing system for packet voice exchange, those skilled  
15 in the art will appreciate that the techniques described for comfort noise generation are likewise suitable for various applications requiring reconstruction of a signal from signal parameters. Accordingly, the described exemplary embodiment for comfort noise generation in a signal processing system for voice applications is by way of example only and not by way of limitation.

20 A comfort noise generator plays noise. In an exemplary embodiment, a comfort noise generator in accordance with ITU standards G.729 Annex B or G.723.1 Annex A may be used. These standards specify background noise levels and spectral content. Referring to FIG. 30, the VAD 1080 in the encoder system determines whether the digital voice samples in the media queue 1066 contain active speech. If the VAD 1080 determines that the digital voice samples  
25 do not contain active speech, then the comfort noise estimator 1081 estimates the energy and spectrum of the background noise parameters at the near end to update a long running background noise energy and spectral estimates. These estimates are periodically quantized and transmitted in a SID packet by the comfort noise estimator (usually at the end of a talk spurt and periodically during the ensuing silent segment, or when the background noise parameters change appreciably). The comfort noise estimator 1081 should update the long running averages, when  
30 necessary, decide when to transmit a SID packet, and quantize and pass the quantized parameters to the packetization engine 1078. SID packets should not be sent while the near end telephony device is on-hook, unless they are required to keep the connection between the telephony devices alive. There may be multiple quantization methods depending on the protocol chosen.

35 In many instances the characterization of spectral content or energy level of the background noise may not be available to the comfort noise generator in the decoder system. For example, SID packets may not be used or the contents of the SID packet may not be specified (see FRF-11). Similarly, the SID packets may only contain an energy estimate, so that estimating

1 some or all of the parameters of the noise in the decoding system may be necessary. Therefore, the comfort noise generator 1092 (see FIG. 30) preferably should not be dependent upon SID packets from the far end encoder system for proper operation.

5 In the absence of SID packets, or SID packets containing energy only, the parameters of the background noise at the far end may be estimated by either of two alternative methods. First, the VAD 1098 at the voice decoder 1096 can be executed in series with the comfort noise estimator 100 to identify silence periods and to estimate the parameters of the background noise during those silence periods. During the identified inactive periods, the digital samples from the voice decoder 1096 are used to update the comfort noise parameters of the comfort noise  
10 estimator. The far end voice encoder should preferably ensure that a relatively long hangover period is used in order to ensure that there are noise-only digital voice samples which the VAD 1098 may identify as inactive speech.

Alternatively, in the case of SID packets containing energy levels only, the comfort noise estimate may be updated with the two or three digital voice frames which arrived immediately  
15 prior to the SID packet. The far end voice encoder should preferably ensure that at least two or three frames of inactive speech are transmitted before the SID packet is transmitted. This can be realized by extending the hangover period. The comfort noise estimator 1100 may then estimate the parameters of the background noise based upon the spectrum and or energy level of these frames. In this alternate approach continuous VAD execution is not required to identify silence  
20 periods, so as to further reduce the average bandwidth required for a typical voice channel.

Alternatively, if it is unknown whether or not the far end voice encoder supports (sending) SID packets, the decoder system may start with the assumption that SID packets are not being sent, utilizing a VAD to identify silence periods, and then only use the comfort noise parameters contained in the SID packets if and when a SID packet arrives.

25 A preferred embodiment of the comfort noise generator generates comfort noise based upon the energy level of the background noise contained within the SID packets and spectral information derived from the previously decoded inactive speech frames. The described exemplary embodiment (in the decoding system) includes a comfort noise estimator for noise analysis and a comfort noise generator for noise synthesis. Preferably there is an extended  
30 hangover period during which the decoded voice samples is primarily inactive before the VAD identifies the signal as being inactive, (changing from speech to noise). Linear Prediction Coding (LPC) coefficients may be used to model the spectral shape of the noise during the hangover period just before the SID packet is received from the VAD. Linear prediction coding models each voice sample as a linear combination of previous samples, that is, as the output of an  
35 all-pole IIR filter. Referring to FIG. 35, a noise analyzer 1174 determines the LPC coefficients.

In the described exemplary embodiment of the comfort noise estimator in the decoding system, a signal buffer 1176 receives and buffers decoded voice samples. An energy estimator 1177 analyzes the energy level of the samples buffered in the signal buffer 1176. The energy

estimator 1177 compares the estimated energy level of the samples stored in the signal buffer with the energy level provided in the SID packet. Comfort noise estimating is terminated if the energy level estimated for the samples stored in the signal buffer and the energy level provided in the SID packet differ by more than a predetermined threshold, preferably on the order of about 6 dB. In addition, the energy estimator 1177, analyzes the stability of the energy level of the samples buffered in the signal buffer. The energy estimator 1177 preferably divides the samples stored in the signal buffer into two groups, (preferably approximately equal halves) and estimates the energy level for each group. Comfort noise estimation is preferably terminated if the estimated energy levels of the two groups differ by more than a predetermined threshold, preferably on the order of about 6 dB. A shaping filter 1178 filters the incoming voice samples from the energy estimator 1177 with a triangular windowing technique. Those of skill in the art will appreciate that alternative shaping filters such as, for example, a Hamming window, may be used to shape the incoming samples.

When a SID packet is received in the decoder system, auto correlation logic 1179 calculates the auto-correlation coefficients of the windowed voice samples. The signal buffer 1176 should preferably be sized to be smaller than the hangover period, to ensure that the auto correlation logic 1179 computes auto correlation coefficients using only voice samples from the hangover period. In the described exemplary embodiment, the signal buffer is sized to store on the order of about two hundred voice samples (25 msec assuming a sample rate of 8000 Hz). Autocorrelation, as is known in the art, involves correlating a signal with itself. A correlation function shows how similar two signals are and how long the signals remain similar when one is shifted with respect to the other. Random noise is defined to be uncorrelated, that is random noise is only similar to itself with no shift at all. A shift of one sample results in zero correlation, so that the autocorrelation function of random noise is a single sharp spike at shift zero. The autocorrelation coefficients are calculated according to the following equation:

$$r(k) = \sum s(n)s(n-k)$$

where  $k=0 \dots p$  and  $p$  is the order of the synthesis filter 1188 (see FIG. 36) utilized to synthesize the spectral shape of the background noise from the LPC filter coefficients.

Filter logic 1180 utilizes the auto correlation coefficients to calculate the LPC filter coefficients 1180(a) and prediction gain 1180(b) using the Levinson-Durbin Recursion method. Preferably, the filter logic 1180 first preferably applies a white noise correction factor to  $r(0)$  to increase the energy level of  $r(0)$  by a predetermined amount. The preferred white noise correction factor is on the order of about  $(257/256)$  which corresponds to a white noise level of approximately 24 dB below the average signal power. The white noise correction factor effectively raises the spectral minima so as to reduce the spectral dynamic range of the auto correlation coefficients to alleviate ill-conditioning of the Levinson-Durbin recursion. As is known in the art, the Levinson-Durbin recursion is an algorithm for finding an all-pole IIR filter with a prescribed deterministic autocorrelation sequence. The described exemplary embodiment

preferably utilizes a tenth order (i.e. ten tap) synthesis filter 1188. However, a lower order filter may be used to realize a reduced complexity comfort noise estimator.

The signal buffer 1176 should preferably be updated each time the voice decoder is invoked during periods of active speech. Therefore, when there is a transition from speech to noise, the buffer 1176 contains the voice samples from the most recent hangover period. The comfort noise estimator should preferably ensure that the LPC filter coefficients is determined using only samples of background noise. If the LPC filter coefficients are determined based on the analysis of active speech samples, the estimated LPC filter coefficients will not give the correct spectrum of the background noise. In the described exemplary embodiment, a hangover period in the range of about 50-250 msec is assumed, and twelve active frames (assuming 5 msec frames) are accumulated before the filter logic 1180 calculates new LPC coefficients.

In the described exemplary embodiment a comfort noise generator utilizes the power level of the background noise retrieved from processed SID packets and the predicted LPC filter coefficients 1180(a) to generate comfort noise in accordance with the following formula:

$$s(n) = e(n) + \sum_{i=1}^M a(i)s(n-i)$$

Where M is the order (i.e. the number of taps) of the synthesis filter 1188,  $s(n)$  is the predicted value of the synthesized noise,  $a(i)$  is the  $i^{\text{th}}$  LPC filter coefficient,  $s(n-i)$  are the previous output samples of the synthesis filter and  $e(n)$  is a Gaussian excitation signal.

A block diagram of the described exemplary embodiment of the comfort noise generator 1182 is shown in FIG. 36. The comfort noise estimator processes SID packets to decode the power level of the current far end background noise. The power level of the background noise is forwarded to a power controller 1184. In addition a white noise generator 1186 forwards a gaussian signal to the power controller 1184. The power controller 1184 adjusts the power level of the gaussian signal in accordance with the power level of the background noise and the prediction gain 1180(b). The prediction gain is the difference in power level of the input and output of synthesis filter 1188. The synthesis filter 1188 receives voice samples from the power controller 1184 and the LPC filter coefficients calculated by the filter logic 1180 (see FIG. 35). The synthesis filter 1188 generates a power adjusted signal whose spectral characteristics approximate the spectral shape of the background noise in accordance with the above equation (i.e. sum of the product of the LPC filter coefficients and the previous output samples of the synthesis filter).

## 5. Voice Encoder/Voice Decoder

The purpose of voice compression algorithms is to represent voice with highest efficiency (i.e., highest quality of the reconstructed signal using the least number of bits). Efficient voice compression was made possible by research starting in the 1930's that demonstrated that voice could be characterized by a set of slowly varying parameters that could later be used to reconstruct an approximately matching voice signal. Characteristics of voice perception allow for lossy compression without perceptible loss of quality.

1 Voice compression begins with an analog-to-digital converter that samples the analog  
voice at an appropriate rate (usually 8,000 samples per second for telephone bandwidth voice)  
and then represents the amplitude of each sample as a binary code that is transmitted in a serial  
fashion. In communications systems, this coding scheme is called pulse code modulation (PCM).

5 When using a uniform (linear) quantizer in which there is uniform separation between  
amplitude levels. This voice compression algorithm is referred to as "linear", or "linear PCM".  
Linear PCM is the simplest and most natural method of quantization. The drawback is that the  
signal-to-noise ratio (SNR) varies with the amplitude of the voice sample. This can be  
10 substantially avoided by using non-uniform quantization known as companded PCM..

In companded PCM, the voice sample is compressed to logarithmic scale before  
transmission, and expanded upon reception. This conversion to logarithmic scale ensures that  
low-amplitude voice signals are quantized with a minimum loss of fidelity, and the SNR is more  
uniform across all amplitudes of the voice sample. The process of compressing and expanding  
15 the signal is known as "companding" (COMpressing and exPANDing). There exists a worldwide  
standard for companded PCM defined by the CCITT (the International Telegraph and Telephone  
Consultative Committee).

The CCITT is a Geneva-based division of the International Telecommunications Union  
(ITU), a New York-based United Nations organization. The CCITT is now formally known as  
20 the ITU-T, the telecommunications sector of the ITU, but the term CCITT is still widely used.  
Among the tasks of the CCITT is the study of technical and operating issues and releasing  
recommendations on them with a view to standardizing telecommunications on a worldwide  
basis. A subset of these standards is the G-Series Recommendations, which deal with the subject  
of transmission systems and media, and digital systems and networks. Since 1972, there have  
25 been a number of G-Series Recommendations on speech coding, the earliest being  
Recommendation G.711. G.711 has the best voice quality of the compression algorithms but the  
highest bit rate requirement.

The ITU-T defined the "first" voice compression algorithm for digital telephony in 1972.  
It is companded PCM defined in Recommendation G.711. This Recommendation constitutes the  
30 principal reference as far as transmission systems are concerned. The basic principle of the  
G.711 companded PCM algorithm is to compress voice using 8 bits per sample, the voice being  
sampled at 8 kHz, keeping the telephony bandwidth of 300-3400 Hz. With this combination,  
each voice channel requires 64 kilobits per second.

35 Note that when the term PCM is used in digital telephony, it usually refers to the  
companded PCM specified in Recommendation G.711, and not linear PCM, since most  
transmission systems transfer data in the companded PCM format. Companded PCM is currently  
the most common digitization scheme used in telephone networks. Today, nearly every



1 telephone call in North America is encoded at some point along the way using G.711 companded PCM.

5 ITU Recommendation G.726 specifies a multiple-rate ADPCM compression technique for converting 64 kilobit per second companded PCM channels (specified by Recommendation G.711) to and from a 40, 32, 24, or 16 kilobit per second channel. The bit rates of 40, 32, 24, and 16 kilobits per second correspond to 5, 4, 3, and 2 bits per voice sample.

10 ADPCM is a combination of two methods: Adaptive Pulse Code Modulation (APCM), and Differential Pulse Code Modulation (DPCM). Adaptive Pulse Code Modulation can be used in both uniform and non-uniform quantizer systems. It adjusts the step size of the quantizer as the voice samples change, so that variations in amplitude of the voice samples, as well as transitions between voiced and unvoiced segments, can be accommodated. In DPCM systems, the main idea is to quantize the difference between contiguous voice samples. The difference is  
15 calculated by subtracting the current voice sample from a signal estimate predicted from previous voice sample. This involves maintaining an adaptive predictor (which is linear, since it only uses first-order functions of past values). The variance of the difference signal results in more efficient quantization (the signal can be compressed coded with fewer bits).

20 The G.726 algorithm reduces the bit rate required to transmit intelligible voice, allowing for more channels. The bit rates of 40, 32, 24, and 16 kilobits per second correspond to compression ratios of 1.6:1, 2:1, 2.67:1, and 4:1 with respect to 64 kilobits per second companded PCM. Both G.711 and G.726 are waveform encoders; they can be used to reduce the bit rate require to transfer any waveform, like voice, and low bit-rate modem signals, while  
25 maintaining an acceptable level of quality.

There exists another class of voice encoders, which model the excitation of the vocal tract to reconstruct a waveform that appears very similar when heard by the human ear, although it may be quite different from the original voice signal. These voice encoders, called vocoders, offer  
30 greater voice compression while maintaining good voice quality, at the penalty of higher computational complexity and increased delay.

For the reduction in bit rate over G.711, one pays for an increase in computational complexity. Among voice encoders, the G.726 ADPCM algorithm ranks low to medium on a  
35 relative scale of complexity, with companded PCM being of the lowest complexity and code-excited linear prediction (CELP) vocoder algorithms being of the highest.

1           The G.726 ADPCM algorithm is a sample-based encoder like the G.711 algorithm, therefore, the algorithmic delay is limited to one sample interval. The CELP algorithms operate on blocks of samples (0.625ms to 30 ms for the ITU coder), so the delay they incur is much greater.

5           The quality of G.726 is best for the two highest bit rates, although it is not as good as that achieved using companded PCM. The quality at 16 kilobits per second is quite poor (a noticeable amount of noise is introduced), and should normally be used only for short periods when it is necessary to conserve network bandwidth (overload situations).

10           The G.726 interface specifies as input to the G.726 encoder (and output to the G.726 decoder) an 8-bit companded PCM sample according to Recommendation G.711. So strictly speaking, the G.726 algorithm is a transcoder, taking log-PCM and converting it to ADPCM, and vice-versa. Upon input of a companded PCM sample, the G.726 encoder converts it to a 14-bit linear PCM representation for intermediate processing. Similarly, the decoder converts an intermediate 14-bit linear PCM value into an 8-bit companded PCM sample before it is output. An extension of the G.726 algorithm was carried out in 1994 to include, as an option, 14-bit linear PCM input signals and output signals. The specification for such a linear interface is given in Annex A of Recommendation G.726.

20           The interface specified by G.726 Annex A bypasses the input and output companded PCM conversions. The effect of removing the companded PCM encoding and decoding is to decrease the coding degradation introduced by the compression and expansion of the linear PCM samples.

25           The algorithm implemented in the described exemplary embodiment can be the version specified in G.726 Annex A, commonly referred to as G.726A, or any other voice compression algorithm known in the art. Among these voice compression algorithms are those standardized for telephony by the ITU-T. Several of these algorithms operate at a sampling rate of 8000 Hz. with different bit rates for transmitting the encoded voice. By way of example, Recommendations G.729 (1996) and G.723.1 (1996) define code excited linear prediction (CELP) algorithms that provide even lower bit rates than G.711 and G.726. G.729 operates at 8 kbps and G.723.1 operates at either 5.3 kbps or 6.3 kbps.

30           In an exemplary embodiment, the voice encoder and the voice decoder support one or more voice compression algorithms, including but not limited to, 16 bit PCM (non-standard, and only used for diagnostic purposes); ITU-T standard G.711 at 64 kb/s; G.723.1 at 5.3 kb/s (ACELP) and 6.3 kb/s (MP-MLQ); ITU-T standard G.726 (ADPCM) at 16, 24, 32, and 40 kb/s; ITU-T standard G.727 (Embedded ADPCM) at 16, 24, 32, and 40 kb/s; ITU-T standard G.728

1 (LD-CELP) at 16 kb/s ; and ITU-T standard G.729 Annex A (CS-ACELP) at 8 kb/s.

5 The packetization interval for 16 bit PCM, G.711, G.726, G.727 and G.728 should be a multiple of 5 msec in accordance with industry standards. The packetization interval is the time duration of the digital voice samples that are encapsulated into a single voice packet. The voice encoder (decoder) interval is the time duration in which the voice encoder (decoder) is enabled. The packetization interval should be an integer multiple of the voice encoder (decoder) interval (a frame of digital voice samples). By way of example, G.729 encodes frames containing 80 digital voice samples at 8 kHz which is equivalent to a voice encoder (decoder) interval of 10 msec. If two subsequent encoded frames of digital voice sample are collected and transmitted in a single packet, the packetization interval in this case would be 20 msec.

15 G.711, G.726, and G.727 encodes digital voice samples on a sample by sample basis. Hence, the minimum voice encoder (decoder) interval is 0.125 msec. This is somewhat of a short voice encoder (decoder) interval, especially if the packetization interval is a multiple of 5 msec. Therefore, a single voice packet will contain 40 frames of digital voice samples. G.728 encodes frames containing 5 digital voice samples (or 0.625 msec). A packetization interval of 5 msec (40 samples) can be supported by 8 frames of digital voice samples. G.723.1 compresses frames containing 240 digital voice samples. The voice encoder (decoder) interval is 30 msec, and the packetization interval should be a multiple of 30 msec.

20 Packetization intervals which are not multiples of the voice encoder (or decoder) interval can be supported by a change to the packetization engine or the depacketization engine. This may be acceptable for a voice encoder (or decoder) such as G.711 or 16 bit PCM.

25 The G.728 standard may be desirable for some applications. G.728 is used fairly extensively in proprietary voice conferencing situations and it is a good trade-off between bandwidth and quality at a rate of 16 kb/s. Its quality is superior to that of G.729 under many conditions, and it has a much lower rate than G.726 or G.727. However, G.728 is MIPS intensive.

30 Differentiation of various voice encoders (or decoders) may come at a reduced complexity. By way of example, both G.723.1 and G.729 could be modified to reduce complexity, enhance performance, or reduce possible IPR conflicts. Performance may be enhanced by using the voice encoder (or decoder) as an embedded coder. For example, the "core" voice encoder (or decoder) could be G.723.1 operating at 5.3 kb/s with "enhancement" information added to improve the voice quality. The enhancement information may be discarded at the source or at any point in the network, with the quality reverting to that of the "core" voice

1 encoder (or decoder). Embedded coders may be readily implemented since they are based on a  
given core. Embedded coders are rate scalable, and are well suited for packet based networks.  
If a higher quality 16 kb/s voice encoder (or decoder) is required, one could use G.723.1 or G.729  
Annex A at the core, with an extension to scale the rate up to 16 kb/s (or whatever rate was  
5 desired).

The configurable parameters for each voice encoder or decoder include the rate at which  
it operates (if applicable), which companding scheme to use, the packetization interval, and the  
core rate if the voice encoder (or decoder) is an embedded coder. For G.727, the configuration  
10 is in terms of bits/sample. For example EADPCM(5,2) (Embedded ADPCM, G.727) has a bit  
rate of 40 kb/s (5 bits/sample) with the core information having a rate of 16 kb/s (2 bits/sample).

## 6. Packetization Engine

15 In an exemplary embodiment, the packetization engine groups voice frames from the  
voice encoder, and with information from the VAD, creates voice packets in a format  
appropriate for the packet based network. The two primary voice packet formats are generic  
voice packets and SID packets. The format of each voice packet is a function of the voice  
encoder used, the selected packetization interval, and the protocol.

20 Those skilled in the art will readily recognize that the packetization engine could be  
implemented in the host. However, this may unnecessarily burden the host with configuration  
and protocol details, and therefore, if a complete self contained signal processing system is  
desired, then the packetization engine should be operated in the network VHD. Furthermore,  
25 there is significant interaction between the voice encoder, the VAD, and the packetization engine,  
which further promotes the desirability of operating the packetization engine in the network VHD

30 The packetization engine may generate the entire voice packet or just the voice portion  
of the voice packet. In particular, a fully packetized system with all the protocol headers may be  
implemented, or alternatively, only the voice portion of the packet will be delivered to the host.  
By way of example, for VoIP, it is reasonable to create the real-time transport protocol (RTP)  
encapsulated packet with the packetization engine, but have the remaining transmission control  
protocol / Internet protocol (TCP/IP) stack residing in the host. In the described exemplary  
35 embodiment, the voice packetization functions reside in the packetization engine. The voice  
packet should be formatted according to the particular standard, although not all headers or all  
components of the header need to be constructed.

## 7. Voice Depacketizing Engine / Voice Queue

In an exemplary embodiment, voice de-packetization and queuing is a real time task which queues the voice packets with a time stamp indicating the arrival time. The voice queue should accurately identify packet arrival time within one msec resolution. Resolution should preferably not be less than the encoding interval of the far end voice encoder. The depacketizing engine should have the capability to process voice packets that arrive out of order, and to dynamically switch between voice encoding methods (i.e. between, for example, G.723.1 and G.711). Voice packets should be queued such that it is easy to identify the voice frame to be released, and easy to determine when voice packets have been lost or discarded en route.

The voice queue may require significant memory to queue the voice packets. By way of example, if G.711 is used, and the worst case delay variation is 250 msec, the voice queue should be capable of storing up to 500 msec of voice frames. At a data rate of 64 kb/s this translates into 4000 bytes or, or 2K (16 bit) words of storage. Similarly, for 16 bit PCM, 500 msec of voice frames require 4K words. Limiting the amount of memory required may limit the worst case delay variation of 16 bit PCM and possibly G.711. This, however, depends on how the voice frames are queued, and whether dynamic memory allocation is used to allocate the memory for the voice frames. Thus, it is preferable to optimize the memory allocation of the voice queue.

The voice queue transforms the voice packets into frames of digital voice samples. If the voice packets are at the fundamental encoding interval of the voice frames, then the delay jitter problem is simplified. In an exemplary embodiment, a double voice queue is used. The double voice queue includes a secondary queue which time stamps and temporarily holds the voice packets, and a primary queue which holds the voice packets, time stamps, and sequence numbers. The voice packets in the secondary queue are disassembled before transmission to the primary queue. The secondary queue stores packets in a format specific to the particular protocol, whereas the primary queue stores the packets in a format which is largely independent of the particular protocol.

In practice, it is often the case that sequence numbers are included with the voice packets, but not the SID packets, or a sequence number on a SID packet is identical to the sequence number of a previously received voice packet. Similarly, SID packets may or may not contain useful information. For these reasons, it may be useful to have a separate queue for received SID packets.

The depacketizing engine is preferably configured to support VoIP, VTOA, VoFR and other proprietary protocols. The voice queue should be memory efficient, while providing the

1 ability to dynamically switch between voice encoders (at the far end), allow efficient reordering  
of voice packets (used for VoIP) and properly identify lost packets.

## 5 8. Voice Synchronization

In an exemplary embodiment, the voice synchronizer analyzes the contents of the voice queue and determines when to release voice frames to the voice decoder, when to play comfort noise, when to perform frame repeats (to cope with lost voice packets or to extend the depth of the voice queue), and when to perform frame deletes (in order to decrease the size of the voice queue). The voice synchronizer manages the asynchronous arrival of voice packets. For those  
10 embodiments which are not memory limited, a voice queue with sufficient fixed memory to store the largest possible delay variation is used to process voice packets which arrive asynchronously. Such an embodiment includes sequence numbers to identify the relative timings of the voice packets. The voice synchronizer should ensure that the voice frames from the voice queue can  
15 be reconstructed into high quality voice, while minimizing the end-to-end delay. These are competing objectives so the voice synchronizer should be configured to provide system trade-off between voice quality and delay.

Preferably, the voice synchronizer is adaptive rather than fixed based upon the worst case  
20 delay variation. This is especially true in cases such as VoIP where the worst case delay variation can be on the order of a few seconds. By way of example, consider a VoIP system with a fixed voice synchronizer based on a worst case delay variation of 300 msec. If the actual delay variation is 280 msec, the signal processing system operates as expected. However, if the actual delay variation is 20 msec, then the end -to-end delay is at least 280 msec greater than required.  
25 In this case the voice quality should be acceptable, but the delay would be undesirable. On the other hand, if the delay variation is 330 msec then an underflow condition could exist degrading the voice quality of the signal processing system.

The voice synchronizer performs four primary tasks. First, the voice synchronizer  
30 determines when to release the first voice frame of a talk spurt from the far end. Subsequent to the release of the first voice frame, the remaining voice frames are released in an isochronous manner. In an exemplary embodiment, the first voice frame is held for a period of time that is equal or less than the estimated worst case jitter.

35 Second, the voice synchronizer estimates how long the first voice frame of the talk spurt should be held. If the voice synchronizer underestimates the required "target holding time," jitter buffer underflow will likely result. However, jitter buffer underflow could also occur at the end of a talk spurt, or during a short silence interval. Therefore, SID packets and sequence numbers

1 could be used to identify what caused the jitter buffer underflow, and whether the target holding  
time should be increased. If the voice synchronizer overestimates the required "target holding  
time," all voice frames will be held too long causing jitter buffer overflow. In response to jitter  
buffer overflow, the target holding time should be decreased. In the described exemplary  
5 embodiment, the voice synchronizer increases the target holding time rapidly for jitter buffer  
underflow due to excessive jitter, but decreases the target holding time slowly when holding  
times are excessive. This approach allows rapid adjustments for voice quality problems while  
being more forgiving for excess delays of voice packets.

10 Thirdly, the voice synchronizer provides a methodology by which frame repeats and  
frame deletes are performed within the voice decoder. Estimated jitter is only utilized to  
determine when to release the first frame of a talk spurt. Therefore, changes in the delay  
variation during the transmission of a long talk spurt must be independently monitored. On  
buffer underflow (an indication that delay variation is increasing), the voice synchronizer  
15 instructs the lost frame recovery engine to issue voice frames repeats. In particular, the frame  
repeat command instructs the lost frame recovery engine to utilize the parameters from the  
previous voice frame to estimate the parameters of the current voice frame. Thus, if frames 1,  
2 and 3 are normally transmitted and frame 3 arrives late, frame repeat is issued after frame  
number 2, and if frame number 3 arrives during this period, it is then transmitted. The sequence  
20 would be frames 1,2, a frame repeat of frame 2 and then frame 3. Performing frame repeats  
causes the delay to increase, which increasing the size of the jitter buffer to cope with increasing  
delay characteristics during long talk spurts. Frame repeats are also issued to replace voice  
frames that are lost en route.

25 Conversely, if the holding time is too large due to decreasing delay variation, the speed  
at which voice frames are released should be increased. Typically, the target holding time can  
be adjusted, which automatically compresses the following silent interval. However, during a  
long talk spurt, it may be necessary to decrease the holding time more rapidly to minimize the  
excessive end to end delay. This can be accomplished by passing two voice frames to the voice  
30 decoder in one decoding interval but only one of the voice frames is transferred to the media  
queue.

The voice synchronizer must also function under conditions of severe buffer overflow,  
where the physical memory of the signal processing system is insufficient due to excessive delay  
35 variation. When subjected to severe buffer overflow, the voice synchronizer could simply  
discard voice frames.

The voice synchronizer should operate with or without sequence numbers, time stamps,

1 and SID packets. The voice synchronizer should also operate with voice packets arriving out of  
order and lost voice packets. In addition, the voice synchronizer preferably provides a variety  
of configuration parameters which can be specified by the host for optimum performance,  
including minimum and maximum target holding time. With these two parameters, it is possible  
5 to use a fully adaptive jitter buffer by setting the minimum target holding time to zero msec and  
the maximum target holding time to 500 msec (or the limit imposed due to memory constraints).  
Although the preferred voice synchronizer is fully adaptive and able to adapt to varying network  
conditions, those skilled in the art will appreciate that the voice synchronizer can also be  
maintained at a fixed holding time by setting the minimum and maximum holding times to be  
10 equal.

### 9. Lost Packet Recovery / Frame Deletion

15 In applications where voice is transmitted through a packet based network there are  
instances where not all of the packets reach the intended destination. The voice packets may  
either arrive too late to be sequenced properly or may be lost entirely. These losses may be  
caused by network congestion, delays in processing or a shortage of processing cycles. The  
packet loss can make the voice difficult to understand or annoying to listen to.

20 Packet recovery refers to methods used to hide the distortions caused by the loss of voice  
packets. In the described exemplary embodiment, a lost packet recovery engine is implemented  
whereby missing voice is filled with synthesized voice using the linear predictive coding model  
of speech. The voice is modelled using the pitch and spectral information from digital voice  
samples received prior to the lost packets.

25 The lost packet recovery engine, in accordance with an exemplary embodiment, can be  
completely contained in the decoder system. The algorithm uses previous digital voice samples  
or a parametric representation thereof, to estimate the contents of lost packets when they occur.

30 FIG. 37 shows a block diagram of the voice decoder and the lost packet recovery engine.  
The lost packet recovery engine includes a voice analyzer 1192, a voice synthesizer 1194 and a  
selector 1196. During periods of no packet loss, the voice analyzer 1192 buffers digital voice  
samples from the voice decoder 1096.

35 When a packet loss occurs, the voice analyzer 1192 generates voice parameters from the  
buffered digital voice samples. The voice parameters are used by the voice synthesizer 1194 to  
synthesize voice until the voice decoder 1096 receives a voice packet, or a timeout period has  
elapsed. During voice syntheses, a "packet lost" signal is applied to the selector to output the  
synthesized voice as digital voice samples to the media queue (not shown).



1 A flowchart of the lost recovery engine algorithm is shown in FIG. 38A. The algorithm  
is repeated every frame, whether or not there has been a lost packet. Every time the algorithm  
is performed, a frame of digital voice samples are output. For purposes of explanation, assume  
a frame length of 5 ms. In this case, forty samples (5 ms of samples for a sampling rate of 8000  
5 Hz) and a flag specifying whether or not there is voice is buffered in the voice analyzer. The  
output of the lost recovery engine is also forty digital voice samples.

First, a check is made to see if there has been a packet loss 1191. If so, then a check is  
made to see if this is the first lost packet in a series of voice packets 1193. If it is the first lost  
packet, then the voice is analysed by calculating the LPC parameters, the pitch, and the voicing  
10 decision 1195 of the buffered digital samples. If the digital samples are voiced 1197, then a  
residual signal is calculated 1199 from the buffered digital voice samples and an excitation signal  
is created from the residual signal 1201. The gain factor for the excitation is set to one. If the  
speech is unvoiced 1197, then the excitation gain factor is determined from a prediction error  
power calculated during a Levinson-Durbin recursion process 1207. Using the parameters  
15 determined from the voice analysis, one frame of voice is synthesized 1201. Finally, the  
excitation gain factor is attenuated 1203, and the synthesized digital voice samples are output  
1205.

If this is not the first lost packet 1193, then a check is made on how many packets have  
been lost. If the number of lost packets exceeds a threshold 1209, then a silence signal is  
20 generated and output 1211. Otherwise, a frame of digital voice samples are synthesized 1201,  
the excitation gain factor is attenuated 1203, and the synthesized digital voice samples are output  
1205.

If there are decoded digital voice samples 1191, then a check is performed to see if there  
was a lost packet the last time the algorithm was executed 1213. If so, then one-half of a frame  
25 of digital voice samples are synthesized, and overlap-added with the first one-half of the frame  
of decoded digital voice samples 1215. Then, in all cases, the digital voice samples are buffered  
in the voice analyser and a frame of digital voice samples is output 1217.

#### a. Calculation of LPC Parameters

30 There are two main steps in finding the LPC parameters. First the autocorrelation  
function  $r(i)$  is determined up to  $r(M)$  where  $M$  is the prediction order. Then the Levinson-  
Durbin recursion formula is applied to the autocorrelation function to get the LPC parameters.

35 There are several steps involved in calculating the autocorrelation function. The  
calculations are performed on the most recent buffered digital voice samples. First, a Hamming  
window is applied to the buffered samples. Then  $r(0)$  is calculated and converted to a floating-  
point format. Next,  $r(1)$  to  $r(M)$  are calculated and converted to floating-point. Finally, a

1 conditioning factor is applied to  $r(0)$  in order to prevent ill conditioning of the  $R$  matrix for a matrix inversion.

5 The calculation of the autocorrelation function is preferably computationally efficient and makes the best use of fixed point arithmetic. The following equation is used as an estimate of the autocorrelation function from  $r(0)$  to  $r(M)$ :

$$r(i) = \sum_{n=0}^{N-i-1} s[n] \cdot s[n-i]$$

where  $s[n]$  is the voice signal and  $N$  is the length of the voice window.

10 The value of  $r(0)$  is scaled such that it is represented by a mantissa and an exponent. The calculations are performed using 16 bit multiplications and the summed results are stored in a 40-bit register. The mantissa is found by shifting the result left or right such that the most significant bit is in bit 30 of the 40-bit register (where the least significant bit is bit 0) and then keeping bits 16 to 31. The exponent is the number of left shifts required for normalization of the mantissa.  
15 The exponent may be negative if a large amplitude signal is present.

The values calculated for  $r(1)$  to  $r(M)$  are scaled to use the same exponent as is used for  $r(0)$ , with the assumption that all values of the autocorrelation function are less than or equal to  $r(0)$ . This representation in which a series of values are represented with the same exponent is called block floating-point because the whole block of data is represented using the same  
20 exponent.

A conditioning factor of 1025/1024 is applied to  $r(0)$  in order to prevent ill conditioning of the  $R$  matrix. This factor increases the value of  $r(0)$  slightly, which has the effect of making  $r(0)$  larger than any other value of  $r(i)$ . It prevents two rows of the  $R$  matrix from having equal values or nearly equal values, which would cause ill conditioning of the matrix. When the matrix  
25 is ill conditioned, it is difficult to control the numerical precision of results during the Levinson-Durbin recursion.

Once the autocorrelation values have been calculated, the Levinson-Durbin recursion formula is applied. In the described exemplary embodiment a sixth to tenth order predictor is  
30 preferably used.

Because of truncation effects caused by the use of fixed point calculations, errors can occur in the calculations when the  $R$  matrix is ill conditioned. Although the conditioning factor applied to  $r(0)$  eliminates this problem for most cases, there is a numerical stability check implemented in the recursion algorithm. If the magnitude of the reflection coefficient gets greater  
35 than or equal to one, then the recursion is terminated, the LPC parameters are set to zero, and the prediction error power is set to  $r(0)$ .

#### b. Pitch Period and Voicing Calculation.

1 The voicing determination and pitch period calculation are performed using the zero crossing count and autocorrelation calculations. The two operations are combined such that the pitch period is not calculated if the zero crossing count is high since the digital voice samples are classified as unvoiced. FIG. 38B shows a flowchart of the operations performed.

5 First the zero crossing count is calculated for a series of digital voice samples 1219. The zero crossing count is initialized to zero. The zero crossings are found at a particular point by multiplying the current digital voice sample by the previous digital voice sample and considering the sign of the result. If the sign is negative, then there was a zero crossing and the zero crossing count is incremented. This process is repeated for a number of digital voice samples, and then  
10 the zero crossing count is compared to a pre-determined threshold. If the count is above the threshold 1221, then the digital voice sample is classified as unvoiced 1223. Otherwise, more computations are performed.

15 Next, if the digital voice samples are not classified as unvoiced, the pitch period is calculated 1225. One way to estimate the pitch period in a given segment of speech is to maximize the autocorrelation coefficient over a range of pitch values. This is shown in equation below:

$$20 \quad P = \arg \max_p \left( \frac{\sum_{i=0}^{N-p-1} s[i] \cdot s[i+p]}{\sqrt{\sum_{i=0}^{N-p-1} s[i] \cdot s[i]} \cdot \sqrt{\sum_{i=0}^{N-p-1} s[i+p] \cdot s[i+p]}} \right)$$

25 An approximation to equation the above equation is used to find the pitch period. First the denominator is approximated by  $r(0)$  and the summation limit in the numerator is made independent of  $p$  as follows

$$30 \quad P = \arg \max_p \left( \frac{\sum_{i=0}^{N-P_{\max}-1} s[i] \cdot s[i+p]}{\sum_{i=0}^{N-P_{\max}-1} s[i] \cdot s[i]} \right)$$

35 where  $p$  is the set of integers greater than or equal to  $P_{\min}$  (preferably on the order of about 20 samples) and less than or equal to  $P_{\max}$  (preferably on the order of about 130 samples). Next, the denominator is removed since it does not depend on  $p$

$$P = \arg \max_p \left( \sum_{i=0}^{N-P_{\max}-1} s[i] \cdot s[i+p] \right)$$

Finally, the speech arrays are indexed such that the most recent samples are emphasized in the estimation of the pitch

$$P = \operatorname{argmax}_p \left( \sum_{i=0}^{N-P_{\max}-1} s[N-1-i] \cdot s[N-1-i-p] \right)$$

This change improves the performance when the pitch is changing in the voice segment under analysis.

When the above equation is applied, a further savings in computations is made by searching only odd values of  $p$ . Once the maximum value has been determined, a finer search is implemented by searching the two even values of  $p$  on either side of the maximum. Although this search procedure is non-optimal, it normally works well because the autocorrelation function is quite smooth for voiced segments.

Once the pitch period has been calculated, the voicing decision is made using the maximum autocorrelation value 1227. If the result is greater than 0.38 times  $r(0)$  then the digital samples are classified as voiced 1229. Otherwise it is classified as unvoiced 1223.

#### c. Excitation Signal Calculation.

For voiced samples, the excitation signal for voice synthesis is derived by applying the following equation to the buffered digital voice samples:

$$e[n] = s[n] - \sum_{i=1}^M a_i \cdot s[n-i]$$

#### d. Excitation Gain Factor for Unvoiced Speech.

For unvoiced samples, the excitation signal for voice synthesis is a white Gaussian noise sequence with a variance of one quarter. In order to synthesize the voice at the correct level, a gain factor is derived from the prediction error power derived during the Levinson-Durbin recursion algorithm. The prediction error power level gives the power level of the excitation signal that will produce a synthesized voice with power level  $r(0)$ . Since a gain level is desired rather than a power level, the square root of the prediction error power level is calculated. To make up for the fact that the Gaussian noise has a power of one quarter, the gain is multiplied by a factor of two.

#### e. Voiced Synthesis.

The voiced synthesis is performed every time there is a lost voiced packet and also for the first decoded voiced packet after a series of lost packets. FIG. 38C shows the steps performed in the synthesis of voice.

1 First, the excitation signal is generated. If the samples are voiced 1231, then the excitation is generated from the residual signal 1233. A residual buffer in the voice analyzer containing the residual signal is modulo addressed such that the excitation signal is equal to repetitions of the past residual signal at the pitch period  $P$ :

$$5 \quad e(n) = \begin{cases} e(n-P) & \text{for } n < P \\ e(n-2P) & \text{for } P \leq n < 2P \\ e(n-3P) & \text{for } 2P \leq n < 3P \\ \dots \end{cases}$$

10 If the value of  $P$  is less than the number of samples to be synthesized, then the excitation signal is repeated more than once. If  $P$  is greater than the number of samples to be generated, then less than one pitch period is contained in the excitation. In both cases the algorithm keeps track of the last index into the excitation buffer such that it can begin addressing at the correct point for the next time voice synthesis is required.

15 If the samples are unvoiced, then a series of Gaussian noise samples are generated 235. Every sample is produced by the addition of twelve uniformly distributed random numbers. Uniformly distributed samples are generated using the linear congruential method (Knuth, 9) as shown by the following equation:

$$20 \quad X_{n+1} = (aX_n + c) \bmod m$$

20 where  $a$  is set to 32763,  $c$  to zero, and  $m$  to 65536. The initial value of  $X_n$  is equal to 29. The sequence of random numbers repeats every 16384 values, which is the maximum period for the chosen value of  $m$  when  $c$  is equal to zero. By choosing  $c$  not equal to zero the period of repetition could be increased to 65536, but 16384 is sufficient for voice synthesis. The longest segment of voice synthesized by the algorithm is twelve blocks of forty samples, which requires  
25 only 5760 uniformly distributed samples. By setting  $c$  to zero, the number of operations to calculate the Gaussian random sample is reduced by one quarter.

30 After the excitation has been constructed, the excitation gain factor is applied to each sample. Finally, the synthesis filter is applied to the excitation to generate the synthetic voice 1237.

#### f. Overlap-Add Calculation.

35 The overlap-add process is performed when the first good packet arrives after one or more lost packets. The overlap-add reduces the discontinuity between the end of the synthesized voice and the beginning of the decoded voice. To overlap the two voice signals, additional digital voice samples (equal to one-half of a frame) is synthesized and averaged with the first one-half frame of the decoded voice packet. The synthesized voice is multiplied by a down-sloping linear ramp and the decoded voice is multiplied by an up-sloping linear ramp. Then the two signals are

1 added together.

## 10. DTMF

5 DTMF (dual-tone, multi-frequency) tones are signaling tones carried within the audio band. A dual tone signal is represented by two sinusoidal signals whose frequencies are separated in bandwidth and which are uncorrelated to avoid false tone detection. A DTMF signal includes one of four tones, each having a frequency in a high frequency band, and one of four tones, each having a frequency in a low frequency band. The frequencies used for DTMF encoding and detection are defined by the ITU and are widely accepted around the world.

10 In an exemplary embodiment of the present invention, DTMF detection is performed by sampling only a portion of each voice frame. This approach results in improved overall system efficiency by reducing the complexity (MIPS) of the DTMF detection. Although the DTMF is described in the context of a signal processing system for packet voice exchange, those skilled in the art will appreciate that the techniques described for DTMF are likewise suitable for various applications requiring signal detection by sampling a portion of the signal. Accordingly, the described exemplary embodiment for DTMF in a signal processing system is by way of example only and not by way of limitation.

20 There are numerous problems involved with the transmission of DTMF in band over a packet based network. For example, lossy voice compression may distort a valid DTMF tone or sequence into an invalid tone or sequence. Also voice packet losses of digital voice samples may corrupt DTMF sequences and delay variation (jitter) may corrupt the DTMF timing information and lead to lost digits. The severity of the various problems depends on the particular voice decoder, the voice decoder rate, the voice packet loss rate, the delay variation, and the particular implementation of the signal processing system. For applications such as VoIP with potentially significant delay variation, high voice packet loss rates, and low digital voice sample rate (if G.723.1 is used), packet tone exchange is desirable. Packet tone exchange is also desirable for VoFR (FRF-11, class 2). Thus, proper detection and out of band transfer via the packet based network is useful.

30 The ITU and Bellcore have promulgated various standards for DTMF detectors. The described exemplary DTMF detector preferably complies with ITU-T Standard Q.24 (for DTMF digit reception) and Bellcore GR-506-Core, TR-TSY-000181, TR-TSY-000762 and TR-TSY-000763, the contents of which are hereby incorporated by reference as though set forth in full herein. These standards involve various criteria, such as frequency distortion allowance, twist allowance, noise immunity, guard time, talk-down, talk-off, acceptable signal to noise ratio, and dynamic range, etc. which are summarized in the table below.

35 The distortion allowance criteria specifies that a DTMF detector should detect a transmitted signal that has a frequency distortion of less than 1.5% and should not detect any

DTMF signals that have frequency distortion of more than 3.5%. The term "twist" refers to the difference, in decibels, between the amplitude of the strongest key pad column tone and the amplitude of the strongest key pad row tone. For example, the Bellcore standard requires the twist to be between -8 and +4 dBm. The noise immunity criteria requires that if the signal has a signal to noise ratio (SNR) greater than certain decibels, then the DTMF detector is required to not miss the signal, i.e., is required to detect the signal. Different standards have different SNR requirements, which usually range from 12 to 24 decibels. The guard time check criteria requires that if a tone has a duration greater than 40 milliseconds, the DTMF detector is required to detect the tone, whereas if the tone has a duration less than 23 milliseconds, the DTMF detector is required to not detect the tone. Similarly, the DTMF detector is required to accept interdigit intervals which are greater than or equal to 40 milliseconds. Alternate embodiments of the present invention readily provide for compliance with other telecommunication standards such as EIA-464B, and JJ-20.12.

Referring to FIG. 39 the DTMF detector 1076 processes the 64kb/s pulse code modulated (PCM) signal, i.e., digital voice samples 1076(a) buffered in the media queue (not shown). The input to the DTMF detector 1076 should preferably be sampled at a rate that is at least higher than approximately 4 kHz or twice the highest frequency of a DTMF tone. If the incoming signal (i.e., digital voice samples) is sampled at a rate that is greater than 4 kHz (i.e. Nyquist for highest frequency DTMF tone) the signal may immediately be downsampled so as to reduce the complexity of subsequent processing. The signal may be downsampled by filtering and discarding samples.

A block diagram of an exemplary embodiment of the invention is shown in FIG. 39. The described exemplary embodiment includes a system for processing the upper frequency band tones and a substantially similar system for processing the lower frequency band tones. A filter 1210 and sampler 1212 may be used to down-sample the incoming signal. In the described exemplary embodiment, the sampling rate is 8 kHz and the front end filter 1210 and sampler 1212 do not down-sample the incoming signal. The output of the sampler 1212 is filtered by two bandpass filters  $H_h(z)$  1214 and  $G_h(z)$  1216 for the upper frequency band and  $H_l(z)$  1218 and  $G_l(z)$  1220 for the lower frequency band) and down-sampled by samplers 1222, 1224 for the upper frequency band and 1226, 1228 for the lower frequency band. The bandpass filters (1214, 1216 and 1218, 1220) for each frequency band are designed using a pair of lowpass filters, one filter  $H(z)$  which multiplies the down-sampled signal by  $\cos(2\pi f_h nT)$  and the other filter  $G(z)$  which multiplies the down-sampled signal by  $\sin(2\pi f_h nT)$  (where  $T = 1/f_s$  where  $f_s$  is the sampling frequency after the front end down-sampling by the filter 1210 and the sampler 1212).

In the described exemplary embodiment, the bandpass filters (1214, 1216 and 1218, 1220) are executed every eight samples and the outputs (1214a, 1216a and 1218a, 1220a) of the bandpass filters (1214, 1216 and 1218, 1220) are down-sampled by samplers 1222, 1224 and 1226, 1228 at a ratio of eight to one. The combination of down-sampling is selected so as to

1 optimize the performance of a particular DSP in use and preferably provides a sample  
 approximately every msec or a 1 kbs signal. Down-sampled signals in the upper and lower  
 frequency bands respectively are real signals. In the upper frequency band, a multiplier 1230  
 multiplies the output of sampler 1224 by the square root of minus one (i.e. j) 1232. A summer  
 1234 then adds the output of downsampler 1222 with the imaginary signal 1230(a). Similarly,  
 5 in the lower frequency band, a multiplier 1236 multiplies the output of downsampler 1228 by the  
 square root of minus one (i.e. j) 1238. A summer 1240 then adds the output of downsampler 226  
 with the imaginary signal 1236(a). Combined signals  $x_h(t)$  1234(a) and  $x_l(t)$  1240(a) at the output  
 of the summers 1234, 1240 are complex signals. It will be appreciated by one of skill in the art  
 that the function of the bandpass filters can be accomplished by alternative finite impulse  
 10 response filters or structures such as windowing followed by DFT processing.

If a single frequency is present within the bands defined by the bandpass filters, the  
 combined complex signals  $x_h(t)$  and  $x_l(t)$  will be constant envelope (complex) signals. Short term  
 power estimator 1242 and 1244 measure the power of  $x_h(t)$  and  $x_l(t)$  respectively and compare  
 15 the estimated power levels of  $x_h(t)$  and  $x_l(t)$  with the requirements promulgated in ITU-T Q.24.  
 In the described exemplary embodiment, the upper band processing is first executed to determine  
 if the power level within the upper band complies with the thresholds set forth in the ITU-T Q.24  
 recommendations. If the power within the upper band does not comply with the ITU-T  
 recommendations the signal is not a DTMF tone and processing is terminated. If the power  
 within the upper band complies with the ITU-T Q.24 standard, the lower band is processed. A  
 20 twist estimator 1246 compares the power in the upper band and the lower band to determine if  
 the twist (defined as the ratio of the power in the lower band and the power in the upper band)  
 is within an acceptable range as defined by the ITU-T recommendations. If the ratio of the power  
 within the upper band and lower band is not within the bounds defined by the standards, a DTMF  
 tone is not present and processing is terminated.

25 If the ratio of the power within the upper band and lower band complies with the  
 thresholds defined by the ITU-T Q.24 and Bellcore GR-506-Core, TR-TSY-000181, TR-TSY-  
 000762 and TR-TSY-000763 standards, the frequency of the upper band signal  $x_h(t)$  and the  
 frequency of the lower band signal  $x_l(t)$  are estimated. Because of the duration of the input signal  
 (one msec), conventional frequency estimation techniques such as counting zero crossings may  
 30 not sufficiently resolve the input frequency. Therefore, differential detectors 1248 and 1250 are  
 used to estimate the frequency of the upper band signal  $x_h(t)$  and the lower band signal  $x_l(t)$   
 respectively. The differential detectors 1248 and 1250 estimate the phase variation of the input  
 signal over a given time range. Advantageously, the accuracy of estimation is substantially  
 insensitive to the period over which the estimation is performed. With respect to upper band  
 input  $x_h(n)$ , (and assuming  $x_h(n)$  is a sinusoid of frequency  $f_i$ ) the differential detector 1248  
 35 computes:

$$y_h(n) = x_h(n)x_h(n-1)*e(-j2\pi f_{mid})$$



1 where  $f_{mid}$  is the mean of the frequencies in the upper band or lower band and superscript\* implies complex conjugation. Then,

$$y_h(n) = e(j2\pi f_i n) e(-j2\pi f_i(n-1)) e(-j2\pi f_{mid}) = e(j2\pi(f_i - f_{mid}))$$

5 which is a constant, independent of n. Arctan functions 1252 and 1254 each takes the complex input and computes the angle of the above complex value that uniquely identifies the frequency present in the upper and lower bands. In operation  $\text{atan2}(\sin(2\pi(f_i - f_{mid})), \cos(2\pi(f_i - f_{mid})))$  returns to within a scaling factor the frequency difference  $f_i - f_{mid}$ . Those skilled in the art will appreciate that various algorithms, such as a frequency discriminator, could be used to  
10 estimate the frequency of the DTMF tone by calculating the phase variation of the input signal over a given time period.

Having estimated the frequency components of the upper band and lower band, the DTMF detector analyzes the upper band and lower band signals to determine whether a DTMF digit is present in the incoming signals and if so which digit. Frequency calculators 1256 and  
15 1258 compute a mean and variance of the frequency deviation over the entire window of frequency estimates to identify valid DTMF tones in the presence of background noise or speech that resembles a DTMF tone. In the described exemplary embodiment, if the mean of the frequency estimates over the window is within acceptable limits, preferably less than  $\pm 2.8\%$  for the lowband and  $\pm 2.5\%$  for the highband the variance is computed. If the variance is less than  
20 a predetermined threshold, preferably on the order of about  $1464 \text{ Hz}^2$  (i.e. standard deviation of  $38.2 \text{ Hz}$ ) the frequency is declared valid. Referring to FIG. 39A, DTMF control logic 1259 compares the frequency identified for the upper and lower bands to the frequency pairs identified in the ITU-T recommendations to identify the digit. The DTMF control logic 1259 forwards a tone detection flag 1259(b) to a state machine 1260. The state machine 1260 analyzes the time  
25 sequence of events and compares the tone on and tone off periods for a given tone to the ITU-T recommendations to determine whether a valid dual tone is present. In the described exemplary embodiment the total window size is preferably 5 msec so that a DTMF detection decision is performed every 5 msec.

In the context of an exemplary embodiment of the voice mode, the DTMF detector is  
30 operating in the packet tone exchange along with a voice encoder operating under the packet voice exchange, which allows for simplification of DTMF detection processing. Most voice encoders operate at a particular frame size (the number of voice samples or time in msec over which voice is compressed). For example, the frame size for ITU-T standard G.723.1 is 30 msec. For ITU-T standard G.729 the frame size is 10 msec. In addition, many packet voice  
35 systems group multiple output frames from a particular voice encoder into a network cell or packet. To prevent leakage through the voice path, the described exemplary embodiment delays DTMF detection until the last frame of speech is processed before a full packet is constructed. Therefore, for transmissions in accordance with the G.723.1 standard and a single output frame placed into a packet, DTMF detection may be invoked every 30 msec (synchronous with the end

1 of the frame). Under the G.729 standard with two voice encoder frames placed into a single packet, DTMF detection or decision may be delayed until the end of the second voice frame within a packet is processed.

5 In the described exemplary embodiment, the DTMF detector is inherently stateless, so that detection of DTMF tones within the second 5 msec DTMF block of a voice encoder frame doesn't depend on DTMF detector processing of the first 5 msec block of that frame. If the delay in DTMF detection is greater than or equal to twice the DTMF detector block size, the processing required for DTMF detection can be further simplified. For example, the instructions required to perform DTMF detection may be reduced by 50% for a voice encoder frame size of 10 msec and a DTMF detector frame size of 5 msec. The ITU-T Q.24 standard requires DTMF tones to have a minimum duration of 23 msec and an inter-digit interval of 40 msec. Therefore, by way of example, a valid DTMF tone may be detected within a given 10 msec frame by only analyzing the second 5 msec interval of that frame. Referring to FIG. 39A, in the described exemplary embodiment, the DTMF control logic 1259 analyzes DTMF detector output 1076(a) and selectively enables DTMF detection analysis 1259(a) for a current frame segment, as a function of whether a valid dual tone was detected in previous and future frame segments. For example, if a DTMF tone was not detected in the previous frame and if DTMF is not present in the second 5 msec interval of the current frame, then the first 5 msec block need not be processed so that DTMF detection processing is reduced by 50%. Similar savings may be realized if the previous frame did contain a DTMF (if the DTMF is still present in the second 5 msec portion it is most likely it was on in the first 5 msec portion). This method is easily extended to the case of longer delays (30 msec for G.723.1 or 20-40 msec for G.729 and packetization intervals from 2-4 or more). It may be necessary to search more than one 5 msec period out of the longer interval, but only a subset is necessary.

25 DTMF events are preferably reported to the host. This allows the host, for example, to convert the DTMF sequence of keys to a destination address. It will, therefore, allow the host to support call routing via DTMF.

30 Depending on the protocol, the packet tone exchange may support muting of the received digital voice samples, or discarding voice frames when DTMF is detected. In addition, to avoid DTMF leakage into the voice path, the voice packets may be queued (but not released) in the encoder system when DTMF is pre-detected. DTMF is pre-detected through a combination of DTMF decisions and state machine processing. The DTMF detector will make a decision (i.e. is there DTMF present) every five msec. A state machine 1260 analyzes the history of a given DTMF tone to determine the current duration of a given tone so as to estimate how long the tone will likely continue. If the detection was false (invalid), the voice packets are ultimately released, otherwise they are discarded. This will manifest itself as occasional jitter when DTMF is falsely pre-detected. It will be appreciated by one of skill in the art that tone packetization can alternatively be accomplished through compliance with various industry standards such as for example, the Frame Relay Forum (FRF -11) standard, the voice over atm standard ITU-T I.363.2,

1 and IETF-draft-avt-tone-04, RTP Payload for DTMF Digits for Telephony Tones and Telephony  
Signals, the contents of which are hereby incorporated by reference as though set forth in full.

5 Software to route calls via DTMF can be resident on the host or within the signal  
processing system. Essentially, the packet tone exchange traps DTMF tones and reports them  
to the host or a higher layer. In an exemplary embodiment, the packet tone exchange will  
generate dial tone when an off-hook condition is detected. Once a DTMF digit is detected, the  
dial tone is terminated. The packet tone exchange may also have to play ringing tone back to the  
near end user (when the far end phone is being rung), and a busy tone if the far end phone is  
10 unavailable. Other tones may also need to be supported to indicate all circuits are busy, or an  
invalid sequence of DTMF digits were entered.

### 11. Call progress tone Detection

15 Telephone systems provide users with feedback about what they are doing in order to  
simplify operation and reduce calling errors. This information can be in the form of lights,  
displays, or ringing, but is most often audible tones heard on the phone line. These tones are  
generally referred to as call progress tones, as they indicate what is happening to dialed phone  
calls. Conditions like busy line, ringing called party, bad number, and others each have distinctive  
tone frequencies and cadences assigned them for which some standards have been established.  
A call progress tone signal includes one of four tones. The frequencies used for call progress  
20 tone encoding and detection, namely 350, 440, 480, and 620 Hz, are defined by the international  
telecommunication union and are widely accepted around the world. The relatively narrow  
frequency separation between tones, 40Hz in one instance complicates the detection of individual  
tones. In addition, the duration or cadence of a given tone is used to identify alternate conditions.

25 An exemplary embodiment of the call progress tone detector analyzes the spectral  
(frequency) characteristics of an incoming telephony voice-band signal and generates a tone  
detection flag as a function of the spectral analysis. The temporal (time) characteristics of the  
tone detection flags are then analyzed to detect call progress tone signals. The call progress tone  
detector then forwards the call progress tone signal to the packetization engine to be packetized  
and transmitted across the packet based network. Although the call progress tone detector is  
30 described in the context of a signal processing system for packet voice exchange, those skilled  
in the art will appreciate that the techniques described for call progress tone detection are  
likewise suitable for various applications requiring signal detection by analyzing spectral or  
temporal characteristics of the signal. Accordingly, the described exemplary embodiment for  
precision tone detection in a signal processing system is by way of example only and not by way  
35 of limitation.

The described exemplary embodiment preferably includes a call progress tone detector  
that operates in accordance with industry standards for the power level (Bellcore SR3004-CPE  
Testing Guidelines; Type III Testing) and cadence (Bellcore GR506-Core and Bellcore LSSGR

1 Signaling For Analog Interface, Call Purpose Signals) of a call progress tone. The call progress  
tone detector interfaces with the media queue to detect incoming call progress tone signals such  
as dial tone, re-order tone, audible ringing and line busy or hook status. The problem of call  
progress tone signaling and detection is a common telephony problem. In the context of packet  
5 voice systems in accordance with an exemplary embodiment of the present invention, telephony  
devices are coupled to a signal processing system which, for the purposes of explanation, is  
operating in a network gateway to support the exchange of voice between a traditional circuit  
switched network and a packet based network. In addition, the signal processing system  
operating on network gateways also supports the exchange of voice between the packet based  
10 network and a number of telephony devices.

10 Referring to FIG. 40 the call progress tone detector 1264 continuously monitors the media  
queue 1066 of the voice encoder system. Typically the call progress tone detector 1264 is  
invoked every ten msec. Thus, for an incoming signal sampled at a rate of 8 kHz, the preferred  
call progress tone detector operates on blocks of eighty samples. The call progress tone detector  
1264 includes a signal processor 1266 which analyzes the spectral characteristics of the samples  
15 buffered in the media queue 1066. The signal processor 1266 performs anti-aliasing, decimation,  
bandpass filtering, and frequency calculations to determine if a tone at a given frequency is  
present. A cadence processor 1268 analyzes the temporal characteristics of the processed tones  
by computing the on and off periods of the incoming signal. If the cadence processor 1268  
detects a call progress tone for an acceptable on and off period in accordance with the Bellcore  
20 GR506-Core standard, a "Tone Detection Event" will be generated.

A block diagram for an exemplary embodiment of the signal processor 1266 is shown in  
FIG. 41. An anti-aliasing low pass filter 1270, with a cutoff frequency of preferably about  
666Hz, filters the samples buffered in the media queue so as to remove frequency components  
above the highest call progress tone frequency, i.e. 660 Hz. A down sampler 1272 is coupled to  
25 the output of the low pass filter 1270. Assuming an 8 kHz input signal, the down sampler 1272  
preferably decimates the low pass filtered signal at a ratio of six:one (which avoids aliasing due  
to under sampling). The output 1272(a) of down sampler 1272 is filtered by eight bandpass  
filters (1274, 1276, 1278, 1280, 1282, 1284, 1286 and 1288), (i.e. two filters for each call  
progress tone frequency). The decimation effectively increases the separation between tones, so  
30 as to relax the roll-off requirements (i.e. reduce the number of filter coefficients) of the bandpass  
filters 1274-1288 which simplifies the identification of individual tones. In the described  
exemplary embodiment, the bandpass filters for each call progress tone 1274-1288 are designed  
using a pair of lowpass filters, one filter which multiplies the down sampled signal by  
 $\cos(2\pi f_n nT)$  and the other filter which multiplies the down sampled signal by  $\sin(2\pi f_n nT)$  (where  
35  $T = 1/f_s$  where  $f_s$  is the sampling frequency after the decimation by the down sampler 1272. The  
outputs of the band pass filters are real signals. Multipliers (1290, 1292, 1294 and 1296)  
multiply the outputs of filters (1276, 1280, 1284 and 1288) respectively by the square root of  
minus one (i.e.  $j$ ) 1298 to generate an imaginary component. Summers (1300, 1302, 1304 and  
1306) then add the outputs of filters (1274, 1278, 1282 and 1286) with the imaginary components

(1290a, 1292a, 1294a and 1296a) respectively. The combined signals are complex signals. It will be appreciated by one of skill in the art that the function of the bandpass filters (1274-1288) can be accomplished by alternative finite impulse response filters or structures such as windowing followed by DFT processing.

Power estimators (1308, 1310, 1312 and 1314) estimate the short term average power of the combined complex signals (1300a, 1302a, 1304a and 1306a) for comparison to power thresholds determined in accordance with the recommended standard (Bellcore SR3004-CPE Testing Guidelines For Type III Testing). The power estimators 1308-1312 forward an indication to power state machines (1316, 1318, 1320 and 1322) respectively which monitor the estimated power levels within each of the call progress tone frequency bands. Referring to FIG. 42, the power state machine is a three state device, including a disarm state 1324, an arm state 1326, and a power on state 1328. As is known in the art, the state of a power state machine depends on the previous state and the new input. For example, if an incoming signal is initially silent, the power estimator 1308 would forward an indication to the power state machine 1316 that the power level is less than the predetermined threshold. The power state machine would be off, and disarmed. If the power estimator 1308 next detects an incoming signal whose power level is greater than the predetermined threshold, the power estimator forwards an indication to the power state machine 1316 indicating that the power level is greater than the predetermined threshold for the given incoming signal. The power state machine 1316 switches to the off but armed state. If the next input is again above the predetermined threshold, the power estimator 1308 forwards an indication to the power state machine 1316 indicating that the power level is greater than the predetermined threshold for the given incoming signal. The power state machine 1316 now toggles to the on and armed state. The power state machine 1316 substantially reduces or eliminates false detections due to glitches, white noise or other signal anomalies.

Turning back to FIG. 41, when the power state machine is set to the on state, frequency calculators (1330, 1332, 1334 and 1336) estimate the frequency of the combined complex signals. The frequency calculators (1330-1336), utilize a differential detection algorithm to estimate the frequency within each of the four call progress tone bands. The frequency calculators (1330-1336) estimate the phase variation of the input signal over a given time range. Advantageously, the accuracy of the estimation is substantially insensitive to the period over which the estimation is performed. Assuming a sinusoidal input  $x(n)$  of frequency  $f_i$  the frequency calculator computes:

$$y(n) = x(n)x(n-1)^*e(-j2\pi f_{mid})$$

where  $f_{mid}$  is the mean of the frequencies within the given call progress tone group and superscript\* implies complex conjugation. Then,

$$y(n) = e(j2\pi f_i n) e(-j2\pi f_i(n-1))e(-j2\pi f_{mid})$$

$$= e(j2\pi(f_i - f_{mid}))$$

which is a constant, independent of  $n$ . The frequency calculators (1330-1336) then invoke an arctan function that takes the complex signal and computes the angle of the above complex value that identifies the frequency present within the given call progress tone band. In operation  $\text{atan2}(\sin(2\pi(f_i - f_{mid})), \cos(2\pi(f_i - f_{mid})))$  returns to within a scaling factor the frequency difference  $f_i - f_{mid}$ . Those skilled in the art will appreciate that various algorithms, such as a frequency discriminator, could be used to estimate the frequency of the call progress tone by calculating the phase variation of the input signal over a given time period.

The frequency calculators (1330-1336) compute the mean of the frequency deviation over the entire 10 msec window of frequency estimates to identify valid call progress tones in the presence of background noise or speech that resembles a call progress tone. If the mean of the frequency estimates over the window is within acceptable limits as summarized by the table below, a tone on flag is forwarded to the cadence processor. The frequency calculators (1330-1336) are preferably only invoked if the power state machine is in the on state thereby reducing the processor loading (i.e. fewer MIPS) when a call progress tone signal is not present.

Tone	Frequency One / Mean	Frequency Two / Mean
Dial Tone	350 Hz / 2 Hz	440 Hz / 3 Hz
Busy	480 Hz / 7 Hz	620 Hz / 9 Hz
Re-order	480 Hz / 7 Hz	620 Hz / 9 Hz
Audible Ringing	440 Hz / 7 Hz	480 Hz / 7 Hz

Referring to FIG. 44, the signal processor 1266 forwards a tone on / tone off indication to the cadence processor 1268 which considers the time sequence of events to determine whether a call progress tone is present. Referring to FIG. 43, in the described exemplary embodiment, the cadence processor 1268 preferably comprises a four state, cadence state machine 1340, including a cadence tone off state 1342, a cadence tone on state 1344, a cadence tone arm state 1346 and an idle state 1348 (see FIG. 43). The state of the cadence state machine 1340 depends on the previous state and the new input. For example, if an incoming signal is initially silent, the signal processor would forward a tone off indication to the cadence state machine 1340. The cadence state machine 1340 would be set to a cadence tone off and disarmed state. If the signal processor next detects a valid tone, the signal processor forwards a tone on indication to the cadence state machine 1340. The cadence state machine 1340 switches to a cadence off but armed state. Referring to FIG. 44, the cadence state machine 1340 preferably invokes a counter 1350 that monitors the duration of the tone indication. If the next input is again a valid call progress tone, the signal processor forwards a tone on indication to the cadence state machine 1340. The cadence state machine 1340 now toggles to the cadence tone on and cadence tone armed state. The cadence state machine 1340 would remain in the cadence tone on state until

receiving two consecutive tone off indications from the signal processor at which time the cadence state machine 1340 sends a tone off indication to the counter 1350. The counter 1350, resets and forwards the duration of the on tone to cadence logic 1352. The cadence processor 1268 similarly estimates the duration of the off tone, which the cadence logic 1352 utilizes to determine whether a particular tone is present by comparing the duration of the on tone, off tone signal pair at a given tone frequency to the tone plan recommended in industry standard as summarized in the table below.

Tone	Duration of Tone On / Tolerance	Duration of Tone Off / Tolerance
Dial Tone	Continuous On	No Off Tone
Busy	500 msec / (+/-50 msec)	500 msec / (+/-50 msec)
Re-order	250 msec / (+/-25 msec)	200 msec / (+/-25 msec)
Audible Ringing	1000 msec / (+/-200 msec)	3000 msec / (+/-2000 msec)
Audible Ringing (Tone 2)	2000 msec / (+/-200 msec)	4000 msec / (+/-2000 msec)

## 12. Resource Manager

In the described exemplary embodiment utilizing a multi-layer software architecture operating on a DSP platform, the DSP server includes networks VHDs (see FIG. 2). Each network VHD can be a complete self-contained software module for processing a single channel with a number of different telephony devices. Multiple channel capability can be achieved by adding network VHDs to the DSP server. The resource manager dynamically controls the creation and deletion of VHDs and services.

In the case of multi-channel communications using a number of network VHDs, the services invoked by the network VHDs and the associated PXDs are preferably optimized to minimize system resource requirements in terms of memory and/or computational complexity. This can be accomplished with the resource manager which reduces the complexity of certain algorithms in the network VHDs based on predetermined criteria. Although the resource management processor is described in the context of a signal processing system for packet voice exchange, those skilled in the art will appreciate that the techniques described for resource management processing are likewise suitable for various applications requiring processor complexity reductions. Accordingly, the described exemplary embodiment for resource management processing in a signal processing system is by way of example only and not by way of limitation.

In one embodiment, the resource manager can be implemented to reduce complexity when the worst case system loading exceeds the peak system resources. The worst case system loading is simply the sum of the worst case (peak) loading of each service invoked by the

1 network VHD and its associated PXDs. However, the statistical nature of the processor resources  
required to process voice band telephony signals is such that it is extremely unlikely that the  
worst case processor loading for each PXD and /or service will occur simultaneously. Thus, a  
more robust ( lower overall power consumption and higher densities, i.e. more channels per DSP)  
5 signal processing system may be realized if the average complexity of the various voice mode  
PXD's and associated services is minimized. Therefore, in the described exemplary embodiment,  
average system complexity is reduced and system resources may be over subscribed (peak  
loading exceeds peak system resources) in the short term wherein complexity reductions are  
invoked to reduce the peak loading placed on the system.

10 The described exemplary resource manager should preferably manage the internal and  
external program and data memory of the DSP. The transmission / signal processing of voice  
is inherently dynamic, so that the system resources required for various stages of a conversation  
are time varying. The resource manager should monitor DSP resource utilization and dynamically  
allocate resources to numerous VHD's and PXD's to achieve a memory and computationally  
15 (reduced MIPS) efficient system. For example, when the near end talker is actively speaking, the  
voice encoder consumes significant resources, but the far end is probably silent so that the echo  
canceller is probably not adapting and may not be executing the transversal filter. When the far  
end is active, the near end is most likely inactive, which implies the echo canceller is both  
canceling far end echo and adapting. However, when the far end is active the near end is  
probably inactive, which implies that the VAD is probably detecting silence and the voice  
20 encoder consumes minimal system resources. Thus, it is unlikely that the voice encoder and echo  
canceller resource utilization peak simultaneously. Furthermore, if processor resources are taxed,  
echo canceller adaptation may be disabled if the echo canceller is adequately adapted or  
interleaved (adaptation enabled on alternating echo canceller blocks) to reduce the computational  
burden placed on the processor.

25 Referring to FIG. 45, in the described exemplary embodiment, the resource manager  
1351 manages the resources of two network VHD's 1062', 1062" and their associated PXD's 1060',  
1060". Initially, the average complexity of the services running in each VHD and its associated  
PXD is reported to the resource manager. The resource manager 1351 sums the reported  
complexities to determine whether the sum exceeds the system resources. If the sum of the  
30 average complexities reported to the resource manager 1351 are within the capability of the  
system resources, no complexity reductions are invoked by the resource manager 1351.  
Conversely, if the sum of the average complexities of the services running in each VHD and its  
associated PXD overload the system resources, then the resource manager can invoke a number  
of complexity reduction methodologies. For example, the echo cancellers 1070', 1070" can be  
35 forced into the bypass mode (see FIG. 31) and/or the echo canceller adaption can be reduced or  
disabled. In addition (or in the alternative), complexity reductions in the voice encoders 1082',  
1082" and voice decoders 1096', 1096" can be invoked.



The described exemplary embodiment may reduce the complexity of certain voice mode services and associated PXDs so as to reduce the computational / memory requirements placed upon the system. Various modifications to the voice encoders may be included to reduce the load placed upon the system resources. For example, the complexity of a G.723.1 voice encoder may be reduced by disabling the post filter in accordance with the ITU-T G.723.1 standard which is incorporated herein by reference as if set forth in full. Also the voicing decision may be modified so as to be based on the open loop normalized pitch correlation computed at the open loop pitch lag L determined by the standard voice encoding algorithm. This entails a modification to the ITU-T G.723.1 C language routine Estim\_Pitch(). If d(n) is the input to the pitch estimation function, the normalized open loop pitch correlation at the open loop pitch lag L is:

$$X(L) = \frac{\sum_{n=0}^{N-1} (d(n) \times d(n-L))^2}{\left( \sum_{n=0}^{N-1} d(n)^2 \right) \left( \sum_{n=0}^{N-1} d(n-L)^2 \right)}$$

where N is equal to a duration of 2 subframes (or 120 samples).

Also, the ability to bypass the adaptive codebook based on a threshold computed from a combination of the open loop normalized pitch correlation and speech/residual energy may be included. In the standard encoder, the search through the adaptive codebook gain codebook begins at index zero and may be terminated before the entire codebook is searched (less than the total size of the adaptive codebook gain codebook which is either 85 or 170 entries) depending on the accumulation of potential error. A preferred complexity reduction truncates the adaptive codebook gain search procedure if the open loop normalized pitch correlation and speech/residual energy meets a certain by searching entries from:

- the upper bound (computed in the standard coder) less half the adaptive codebook size (or index zero, whichever is greater) for voiced speech; and

- from index zero up to half the size of the adaptive code gain codebook (85/2 or 170/2).

The adaptive codebook may also be completely bypassed under some conditions by setting the adaptive codebook gain index to zero, which selects an all zero adaptive codebook gain setting.

The fixed excitation in the standard encoder may have a periodic component. In the standard encoder, if the open loop pitch lag is less than the subframe length minus two, then a excitation search function (the function call Find\_Best() in the ITU-T G.723.1 C language simulation) is invoked twice. To reduce system complexity, the fixed excitation search procedure may be modified (at 6.3 kb/s) such that the fixed excitation search function is invoked once per

1 invocation of the fixed excitation search procedure (routine Find\_Fcbk()). If the open loop pitch  
lag is less than the subframe length minus two then a periodic repetition is forced, otherwise there  
is no periodic repetition (as per the standard encoder for that range of open loop pitch lags). In  
the described complexity reduction modification, the decision on which manner to invoke it is  
5 based on the open loop pitch lag and the voicing strength.

Similarly, the fixed excitation search procedure can be modified (at 5.3 kb/s) such that  
a higher threshold is chosen for voice decisions. In the standard encoder, the voicing decision is  
considered to be voiced if the open loop normalized pitch correlation is greater than 0.5 (variable  
named "threshold" in the ITU-T G.723.1) is set to 0.5. In a modification to reduce the complexity  
10 of this function, the threshold may be set to 0.75. This greatly reduces the complexity of the  
excitation search procedure while avoiding substantial impairment to the voice quality.

Similar modifications may be made to reduce the complexity of a G.729 Annex A voice  
encoder. For example, the complexity of a G.729 Annex A voice encoder may be reduced by  
15 disabling the post filter in accordance with the G.729 Annex A standard which is incorporated  
herein by reference as if set out in full. Also, the complexity of a G.729 Annex A voice encoder  
may be further reduced by including the ability to bypass the adaptive codebook or reduce the  
complexity of the adaptive codebook search significantly. In the standard voice encoder, the  
adaptive codebook searches over a range of lags based on the open loop pitch lag. The adaptive  
codebook bypass simply chooses the minimum lag. The complexity of the adaptive codebook  
20 search may be reduced by truncating the adaptive codebook search such that fractional pitch  
periods are not considered within the search (not searching the non-integer lags). These  
modifications are made to the ITU-T G.729 Annex A, C language routine Pitch\_fr3\_fast(). The  
complexity of a G.729 Annex A voice encoder may be further reduced by substantially reducing  
the complexity of the fixed excitation search. The search complexity may be reduced by  
25 bypassing the depth first search 4, phase A: track 3 and 0 search and the depth first search 4,  
phase B: track 1 and 2 search.

Each modification reduces the computational complexity but also minimally reduces the  
resultant voice quality. However, since the voice encoders are externally managed by the  
resource manager to minimize occasional system resource overloads, the voice encoder should  
30 predominately operate with no complexity reductions. The preferred embedded software  
embodiment should include the standard code as well as the modifications required to reduce the  
system complexity. The resource manager should preferably minimize power consumption and  
computational cycles by invoking complexity reductions which have substantially no impact on  
voice quality. The different complexity reductions schemes should be selected dynamically  
35 based on the processing requirements for the current frame (over all voice channels) and the  
statistics of the voice signals on each channel (voice level, voicing, etc).

Although complexity reductions are rare, the appropriate PxDs and associated services  
invoked in the network VHDs should preferably incorporate numerous functional features to

1 accommodate such complexity reductions. For example, the appropriate voice mode PXDs and  
associated services should preferably include a main routine which executes the complexity  
reductions described above with a variety of complexity levels. For example, various complexity  
5 levels may be mandated by setting various complexity reduction flags. In addition, the resource  
manager should accurately measure the resource requirements of PXDs and services with fixed  
resource requirements (i.e. complexity is not controllable), to support the computation of peak  
complexity and average complexity. Also, a function that returns the estimated complexity in  
cycles according to the desired complexity reduction level should preferably be included.

10 The described exemplary embodiment preferably includes four complexity reduction  
levels. In the first level, all complexity reductions are disabled so that the complexity of the  
PXDs and services is not reduced.

15 The second level provides minimal or transparent complexity reductions (reductions  
which should preferably have substantially no observable impact on performance under most  
conditions). In the transparent mode the voice encoders (G.729, G.723.1) preferably use  
voluntary reductions and the echo canceller is forced into the bypass mode and adaption is  
20 toggled (i.e., adaptive is enabled for every other frame). Voluntary reductions for G.723.1 voice  
encoders are preferably selected as follows. First, if the frame energy is less than -55 dBm0, then  
the adaptive codebook is bypassed and the fixed excitation searches are reduced, as per above.  
If the frame energy is less than -45 dBm0 but greater than -55 dBm0, then the adaptive codebook  
is partially searched and the fixed excitation searches are reduced as per above. In addition, if  
25 the open loop normalized pitch correlation is less than 0.305 then the adaptive codebook is  
partially searched. Otherwise, no complexity reductions are done. Similarly, voluntary  
reductions for the G.729 voice encoders preferably proceed as follows: first, if the frame energy  
is less than -55 dBm0, then the adaptive codebook is bypassed and the fixed excitation search is  
reduced per above. Next if the frame energy is less than -45 dBm0 but greater than -55 dBm0,  
30 then the reduced complexity adaptive codebook is used and the excitation search complexity is  
reduced. Otherwise, no complexity reduction is used.

35 The third level of complexity reductions provides minor complexity reductions  
(reductions which may result in a slight degradation of voice quality or performance). For  
example, in the third level the voice encoders preferably use voluntary reductions, "find\_best"  
reduction (G.723.1), fixed codebook threshold change (5.3 kbps G.723.1), open loop pitch search  
reduction (G.723.1 only), and minimal adaptive codebook reduction (G.729 and G.723.1). In  
addition, the echo canceller is forced into the bypass mode and adaption is toggled.

40 In the fourth level major complexity reductions occur, that is reductions which should  
noticeably effect the performance quality. For example, in the fourth level of complexity  
reductions the voice encoders use the same complexity reductions as those used for level three  
reductions, as well as adding a bypass adaptive codebook reduction (G.729 and G.723.1). In  
addition, the echo canceller is forced into the bypass mode and adaption is completely disabled.

1 The resource manager preferably limits the invocation of fourth level major reductions to extreme circumstances, such as, for example when there is double talk on all active channels.

5 The described exemplary resource manager monitors system resource utilization. Under normal system operating conditions, complexity reductions are not mandated on the echo canceller or voice encoders. Voice/FAX and data traffic is packetized and transferred in packets. The echo canceller removes echos, the DTMF detector detects the presence of keypad signals, the VAD detects the presence of voice, and the voice encoders compress the voice traffic into packets. However, when system resources are overtaxed and complexity reductions are required there are at least two methods for controlling the voice encoder. In the first method, the complexity level for the current frame is estimated from the information contained within previous voice frames and from the information gained from the echo canceller on the current voice frame. The resource manager then mandates complexity reductions for the processing of frames in the current frame interval in accordance with these estimations.

15 Alternatively, the voice encoders may be divided into a "front end" and a "back end". The front end performs voice activity detection and open loop pitch detection (in the case of G.723.1 and G.729 Annex A) on all channels operating on the DSP. Subsequent to the execution of the front end function for all channels of a particular voice encoder, the system complexity may be estimated based on the known information. Complexity reductions may then be mandated to ensure that the current processing cycle can satisfy the processing requirements of the voice encoders and decoders. This alternative method is preferred because the state of the VAD is known whereas in the previously described method the state of the VAD is estimated.

25 In the alternate method, once the front end processing is complete so that the state of the VAD and the voicing state for all channels is known, the system complexity may be estimated based on the known statistics for the current frame. In the first method, the state of the VAD and the voicing state may be estimated based on available known information. For example, the echo canceller processes a voice encoder input signal to remove line echos prior to the activation of the voice encoder. The echo canceller may estimate the state of the VAD based on the power level of a reference signal and the voice encoder input signal so that the complexity level of all controllable PXDs and services may be updated to determine the estimated complexity level of each assuming no complexity reductions have been invoked. If the sum of all the various complexity estimates is less than the complexity budget, no complexity reductions are required. Otherwise, the complexity level of all system components are estimated assuming the invocation of the transparent complexity reduction method to determine the estimated complexity resources required for the current processing frame. If the sum of the complexity estimates with transparent complexity reductions in place is less than the complexity budget, then the transparent complexity reduction is used for that frame. In a similar manner, more and more severe complexity reduction is considered until system complexity satisfies the prescribed budget.

1       The operating system should preferably allow processing to exceed the real-time  
constraint, i.e. maximum processing capability for the underlying DSP, in the short term. Thus  
data that should normally be processed within a given time frame or cycle may be buffered and  
processed in the next sequence. However, the overall complexity or processor loading must  
5       remain (on average) within the real-time constraint. This is a tradeoff between delay/jitter and  
channel density. Since packets may be delayed (due to processing overruns) overall end to end  
delay may increase slightly to account for the processing jitter.

10       Referring to FIG. 31, a preferred echo canceller has been modified to include an echo  
canceller bypass switch that invokes an echo suppressor in lieu of echo cancellation under certain  
system conditions so as to reduce processor loading. In addition, in the described exemplary  
embodiment the resource manager may instruct the adaptation logic 1136 to disable filter adapter  
1134 so as to reduce processor loading under real-time constraints. The system will preferably  
limit adaptation on a fair and equitable basis when processing overruns occur. For example, if  
15       four echo cancellers are adapting when a processing over run occurs, the resource manager may  
disable the adaption of echo cancellers one and two. If the processing over run continues, the  
resource manger should preferably enable adaption of echo cancellers one and two, and reduce  
system complexity by disabling the adaptation of echo cancellers three and four. This limitation  
should preferably be adjusted such that channels which are fully adapted have adaptation disabled  
first. In the described exemplary embodiment, the operating systems should preferably control  
20       the subfunctions to limit peak system complexity. The subfunctions should be co-operative and  
include modifications to the echo canceller and the speech encoders.

#### B.    The Fax Relay Mode

25       Fax relay mode provides signal processing of fax signals. Fax relay mode enables the  
transmission of fax signals over a packet based system such as VoIP, VoFR, FRF-11, VTOA,  
or any other proprietary network. For the purposes of explanation, first fax machine is called a  
sending fax that is connected to the sending network gateway 1378a through a PSTN. The  
sending network gateway is connected to a CMTS via a HFC network. Additional fax machines  
may be on line connections coupled to the other end of the CMTS via a network gateway and a  
HFC network, or off line connections, coupled to the CMTS for example by a telephone network  
30       gateway and a PSTN.

35       The transfer of fax signals over packet based networks may be accomplished by at least  
three alternative methods. In the first method, fax data signals are exchanged in real time.  
Typically, the sending and receiving fax machines are spoofed to allow transmission delays plus  
jitter of up to about 1.2 seconds. The second, store and forward mode, is a non real time method  
of transferring fax data signals. Typically, the fax communication is transacted locally, stored  
into memory and transmitted to the destination fax machine at a subsequent time. The third  
mode is a combination of store and forward mode with minimal spoofing to provide an  
approximate emulation of a typical fax connection.

1 In the fax relay mode, the network VHD invokes the packet fax data exchange. The  
packet fax data exchange provides demodulation and re-modulation of fax data signals. This  
approach results in considerable bandwidth savings since only the underlying unmodulated data  
signals are transmitted across the packet based network. The packet fax data exchange also  
5 provides compensation for network jitter with a jitter buffer similar to that invoked in the packet  
voice exchange. Additionally, the packet fax data exchange compensates for lost data packets  
with error correction processing. Spoofing may also be provided during various stages of the  
procedure between the fax machines to keep the connection alive.

10 The packet fax data exchange is divided into two basic functional units, a demodulation  
system and a re-modulation system. In the demodulation system, the network VHD couples fax  
data signals from a circuit switched network, or a fax machine, to the packet based network. In  
the re-modulation system, the network VHD couples fax data signals from the packet network  
to the switched circuit network, or a fax machine directly.

15 During real time relay of fax data signals over a packet based network, the sending and  
receiving fax machines are spoofed to accommodate network delays plus jitter. Typically, the  
packet fax data exchange can accommodate a total delay of up to about 1.2 seconds. Preferably,  
the packet fax data exchange supports error correction mode (ECM) relay functionality, although  
a full ECM implementation is typically not required. In addition, the packet fax data exchange  
should preferably preserve the typical call duration required for a fax session over a PSTN/ISDN  
20 when exchanging fax data signals between two terminals.

25 The packet fax data exchange for the real time exchange of fax data signals between a  
circuit switched network and a packet based network is shown schematically in FIG. 46. In this  
exemplary embodiment, a connecting PXD (not shown) connecting the fax machine to the switch  
board 1032' is transparent, although those skilled in the art will appreciate that various signal  
conditioning algorithms could be programmed into PXD such as echo cancellation and gain.

30 After the PXD (not shown), the incoming fax data signal 1390a is coupled to the  
demodulation system of the packet fax data exchange operating in the network VHD via the  
switchboard 1032'. The incoming fax data signal 1390a is received and buffered in an ingress  
media queue 1390. A V.21 data pump 1392 demodulates incoming T.30 message so that T.30  
relay logic 1394 can decode the received T.30 messages 1394a. Local T.30 indications 1394b  
are packetized by a packetization engine 1396 and if required, translated into T.38 packets via  
a T.38 shim 1398 for transmission to a T.38 compliant remote network gateway (not shown)  
across the packet based network. The V.21 data pump 1392 is selectively enabled/disabled  
1394c by the T.30 relay logic 1394 in accordance with the reception/ transmission of the T.30  
35 messages or fax data signals. The V.21 data pump 1392 is common to the demodulation and re-  
modulation system. The V.21 data pump 1392 communicates T.30 messages such as for  
example called station tone (CED) and calling station tone (CNG) to support fax setup between

1 a local fax device (not shown) and a remote fax device (not shown) via the remote network gateway.

5 The demodulation system further includes a receive fax data pump 1400 which demodulates the fax data signals during the data transfer phase. The receive fax data pump 1400 supports the V.27ter standard for fax data signal transfer at 2400/4800 bps, the V.29 standard for fax data signal transfer at 7200/9600 bps, as well as the V.17 standard for fax data signal transfer at 7200/9600/12000/14400 bps. The V.34 fax standard, once approved, may also be supported. The T.30 relay logic 1394 enables / disables 1394d the receive fax data pump 1400 in accordance with the reception of the fax data signals or the T.30 messages.

10 If error correction mode (ECM) is required, receive ECM relay logic 1402 performs high level data link control( HDLC )de-framing, including bit de-stuffing and preamble removal on ECM frames contained in the data packets. The resulting fax data signals are then packetized by the packetization engine 1396 and communicated across the packet based network. The T.30 relay logic 1394 selectively enables / disables 1394e the receive ECM relay logic 1402 in accordance with the error correction mode of operation.

15 In the re-modulation system, if required, incoming data packets are first translated from a T.38 packet format to a protocol independent format by the T.38 packet shim 1398. The data packets are then de-packetized by a depacketizing engine 1406. The data packets may contain T.30 messages or fax data signals. The T.30 relay logic 1394 reformats the remote T.30 indications 1394f and forwards the resulting T.30 indications to the V.21 data pump 1392. The modulated output of the V.21 data pump 1392 is forwarded to an egress media queue 1408 for transmission in either analog format or after suitable conversion, as 64 kbps PCM samples to the local fax device over a circuit switched network, such as for example a PSTN line.

20 De-packetized fax data signals are transferred from the depacketizing engine 1406 to a jitter buffer 1410. If error correction mode (ECM) is required, transmitting ECM relay logic 1412 performs HDLC de-framing, including bit stuffing and preamble addition on ECM frames. The transmitting ECM relay logic 1412 forwards the fax data signals, (in the appropriate format) to a transmit fax data pump 1414 which modulates the fax data signals and outputs 8 KHz digital samples to the egress media queue 1408. The T.30 relay logic selectively enables/disables (1394g) the transmit ECM relay logic 1412 in accordance with the error correction mode of operation.

25 The transmit fax data pump 1414 supports the V.27ter standard for fax data signal transfer at 2400/4800 bps, the V.29 standard for fax data signal transfer at 7200/9600 bps, as well as the V.17 standard for fax data signal transfer at 7200/9600/12000/14400 bps. The T.30 relay logic selectively enables/disables (1394h) the transmit fax data pump 1414 in accordance with the transmission of the fax data signals or the T.30 message samples.

1           If the jitter buffer 1410 underflows, a buffer low indication 1410a is coupled to spoofing  
logic 1416. Upon receipt of a buffer low indication during the fax data signal transmission, the  
spoofing logic 1416 inserts "spoofed data" at the appropriate place in the fax data signals via the  
transmit fax data pump 1414 until the jitter buffer 1410 is filled to a pre-determined level, at  
5           which time the fax data signals are transferred out of the jitter buffer 1410. Similarly, during the  
transmission of the T.30 message indications, the spoofing logic 1416 can insert "spoofed data"  
at the appropriate place in the T.30 message samples via the V.21 data pump 1392.

#### 1.       Data Rate Management

10           An exemplary embodiment of the packet fax data exchange complies with the T.38  
recommendations for real-time Group 3 facsimile communication over packet based networks.  
In accordance with the T.38 standard, the preferred system should therefore, provide packet fax  
data exchange support at both the T.30 level (see ITU Recommendation T.30 - "Procedures for  
Document Facsimile Transmission in the General Switched Telephone Network", 1988) and the  
15           T4 level (see ITU Recommendation T.4 - "Standardization of Group 3 Facsimile Apparatus For  
Document Transmission", 1998), the contents of each of these ITU recommendations being  
incorporated herein by reference as if set forth in full. One function of the packet fax data  
exchange is to relay the set up (capabilities) parameters in a timely fashion. Spoofing may be  
needed at either or both the T.30 and T.4 levels to maintain the fax session while set up  
parameters are negotiated at each of the network gateways and relayed in the presence of network  
20           delays and jitter.

          In accordance with the industry T.38 recommendations for real time Group 3  
communication over packet based networks, the described exemplary embodiment relays all  
information including; T.30 preamble indications (flags), T.30 message data, as well as T.30  
25           image data between the network gateways. The T.30 relay logic 1394 in the sending and  
receiving network gateways then negotiate parameters as if connected via a PSTN line. The  
T.30 relay logic 1394 interfaces with the V.21 data pump 1392 and the receive and transmit data  
pumps 1400 and 1414 as well as the packetization engine 1396 and the depacketizing engine  
1406 to ensure that the sending and the receiving fax machines 1380(a) and 1380(b) successfully  
and reliably communicate. The T.30 relay logic 1394 provides local spoofing, using command  
30           repeats (CRP), and automatic repeat request (ARQ) mechanisms, incorporated into the T.30  
protocol, to handle delays associated with the packet based network. In addition, the T.30 relay  
logic 1394 intercepts control messages to ensure compatibility of the rate negotiation between  
the near end and far end machines including HDLC processing, as well as lost packet recovery  
according to the T.30 ECM standard.

35           FIG. 47 demonstrates message flow over a packet based network between a sending fax  
machine 1380a (see FIG. 45) and the receiving fax device 1380b (see FIG. 45) in non-ECM  
mode. The PSTN fax call is divided into five phases: call establishment, control and capabilities  
exchange, page transfer, end of page and multi-page signaling and call release. In the call



1 establishment phase, the sending fax machine dials the sending network gateway 1378a (see FIG. 45) which forwards calling tone (CNG) (not shown) to the receiving network gateway 1378b (see FIG. 45). The receiving network gateway responds by alerting the receiving fax machine. The receiving fax machine answers the call and sends called station (CED) tones. The CED tones  
5 are detected by the V.21 data pump 1392 of the receiving network gateway which issues an event 1420 indicating the receipt of CED which is then relayed to the sending network gateway. The sending network gateway forwards the CED tone 1422 to the sending fax device. In addition, the V.21 data pump of the receiving network gateway invokes the packet fax data exchange.

10 In the control and capabilities exchange, the receiving network gateway transmits T.30 preamble (HDLC flags) 1424 followed by called subscriber identification (CSI) 1426 and digital identification signal (DIS) 1428 message which contains the capabilities of the receiving fax device. The sending network gateway, forwards the HDLC flags, CSI and DIS to the sending fax device. Upon receipt of CSI and DIS, the sending fax device determines the conditions for the call by examining its own capabilities table relative to those of the receiving fax device. The  
15 sending fax device issues a command to the sending network gateway 1430 to begin transmitting HDLC flags. Next, the sending fax device transmits subscriber identification (TSI) 1432 and digital command signal (DCS) 1434 messages, which define the conditions of the call to the sending network gateway. In response, the sending network gateway forwards V.21 HDLC sending subscriber identification / frame check sequences and digital command signal / frame check sequences to the receiving fax device via the receiving network gateway. Next the sending  
20 fax device transmits training check (TCF) fields 1436 to verify the training and ensure that the channel is suitable for transmission at the accepted data rate.

The TCF 1436 may be managed by one of two methods. The first method, referred to as the data rate management method one in the T.38 standard, the receiving network gateway locally  
25 generate TCF. Confirmation to receive (CFR) is returned to the sending fax device 1380(a), when the sending network gateway receives a confirmation to receive (CFR) 1438 from the receiving fax machine via the receiving network gateway, and the TCF training 1436 from the sending fax machine is received successfully. In the event that the receiving fax machine receives a CFR and the TCF training 1436 from the sending fax machine subsequently fails, then DCS 1434 from the sending fax machine is again relayed to the receiving fax machine. The TCF  
30 training 1436 is repeated until an appropriate rate is established which provides successful TCF training 1436 at both ends of the network.

In a second method to synchronize the data rate, referred to as the data rate management method two in the T.38 standard, the TCF data sequence received by the sending network  
35 gateway is forwarded from the sending fax machine to the receiving fax machine via the receiving network gateway. The sending and receiving fax machines then perform speed selection as if connected via a regular PSTN.

1           Upon receipt of confirmation to receive (CFR) 1440 which indicates that all capabilities  
and the modulation speed have been confirmed, the sending fax machine enters the page transfer  
phase, and transmits image data 1444 along with its training preamble 1442. The sending  
network gateway receives the image data and forwards the image data 1444 to the receiving  
5           network gateway. The receiving network gateway then sends its own training preamble 1446  
followed by the image data 1448 to the receiving fax machine.

10           In the end of page and multi-page signaling phase, after the page has been successfully  
transmitted, the sending fax device sends an end of procedures (EOP) 1450 message if the fax  
call is complete and all pages have been transmitted. If only one of multiple pages has been  
successfully transmitted, the sending fax device transmits a multi-page signal (MPS). The  
receiving fax device responds with message confirmation (MCF) 1452 to indicate the message  
has been successfully received and that the receiving fax device is ready to receive additional  
pages. The release phase is the final phase of the call, where at the end of the final page, the  
receiving fax machine sends a message confirmation (MCF) 1452, which prompts the sending  
15           fax machine to transmit a disconnect (DCN) signal 1454. The call is then terminated at both  
ends of the network.

20           ECM fax relay message flow is similar to that described above. All preambles, messages  
and page transfers (phase C) HDLC data are relayed through the packet based network. Phase  
C HDLC data is de-stuffed and, along with the preamble and frame checking sequences (FCS),  
removed before being relayed so that only fax image data itself is relayed over the packet based  
network. The receiving network gateway performs bit stuffing and reinserts the preamble and  
FCS.

## 2.     Spoofing Techniques

25           Spoofing refers to the process by which a facsimile transmission is maintained in the  
presence of data packet under-run due to severe network jitter or delay. An exemplary  
embodiment of the packet fax data exchange complies with the T.38 recommendations for real-  
time Group 3 facsimile communication over packet based networks. In accordance with the T.38  
recommendations, a local and remote T.30 fax device communicate across a packet based  
30           network via signal processing systems, which for the purposes of explanation are operating in  
network gateways. In operation, each fax device establishes a facsimile connection with its  
respective network gateway in accordance with the ITU-T.30 standards and the signal processing  
systems operating in the network gateways relay data signals across a packet based network.

35           In accordance with the T.30 protocol, there are certain time constraints on the  
handshaking and image data transmission for the facsimile connection between the T.30 fax  
device and its respective network gateway. The problem that arises is that the T.30 facsimile  
protocol is not designed to accommodate the significant jitter and packet delay that is common  
to communications across packet based networks. To prevent termination of the fax connection

1 due to severe network jitter or delay, it is, therefore, desirable to ensure that both T.30 fax devices  
can be spoofed during periods of data packet under-run. FIG. 48 demonstrates fax  
communication 1466 under the T.30 protocol, wherein a handshake negotiator 1468, typically  
a low speed modem such as V.21, performs handshake negotiation and fax image data is  
5 communicated via a high speed data pump 1470 such as V.27, V.29 or V.17. In addition, fax  
image data can be transmitted in an error correction mode (ECM) 1472 or non error correction  
mode (non-ECM) 1474, each of which uses a different data format.

Therefore, in the described exemplary embodiment, the particular spoofing technique  
utilized is a function of the transmission format. In the described exemplary embodiment, HDLC  
10 preamble 1476 is used to spoof the T.30 fax devices during V.21 handshaking and during  
transmission of fax image data in the error correction mode. However, zero-bit filling 1478 is  
used to spoof the T.30 fax devices during fax image data transfer in the non error correction  
mode. Although fax relay spoofing is described in the context of a signal processing system with  
the packet data fax exchange invoked, those skilled in the art will appreciate that the described  
15 exemplary fax relay spoofing method is likewise suitable for various other telephony and  
telecommunications application. Accordingly, the described exemplary embodiment of fax relay  
spoofing in a signal processing system is by way of example only and not by way of limitation.

a. V.21 HDLC Preamble Spoofing

20 The T.30 relay logic 1394 packages each message or command into a HDLC frame which  
includes preamble flags. An HDLC frame structure is utilized for all binary-coded V.21  
facsimile control procedures. The basic HDLC structure consists of a number of frames, each  
of which is subdivided into a number of fields. The HDLC frame structure provides for frame  
labeling and error checking. When a new facsimile transmission is initiated, HDLC preamble  
25 in the form of synchronization sequences are transmitted prior to the binary coded information.  
The HDLC preamble is V.21 modulated bit streams of "01111110 (0x7e)".

In the described exemplary embodiment, spoofing techniques are utilized at the T.30 and  
T.4 levels to manage extended network delays and jitter. Turning back to FIG. 46, the T.30 relay  
30 logic 1394 waits for a response to any message or command transmitted across the packet based  
network before continuing to the next state or phase. In accordance with an exemplary spoofing  
technique, the sending and receiving network gateways 1378a, 1378b (See FIG. 45) spoof their  
respective fax machines 1380a, 1380b by locally transmitting HDLC preamble flags if a response  
to a transmitted message is not received from the packet based network within approximately 1.5-  
2.0 seconds. The maximum length of the preamble is limited to about four seconds. If a  
35 response from the packet based network arrives before the spoofing time out, each network  
gateway should preferably transmit a response message to its respective fax machine following  
the preamble flags. Otherwise, if the network response to a transmitted message is not received  
prior to the spoofing time out (in the range of about 5.5-6.0 seconds), the response is assumed

1 to be lost. In this case, when the network gateway times out and terminates preamble spoofing, the local fax device transmits the message command again. Each network gateway repeats the spoofing technique until a successful handshake is completed or its respective fax machine disconnects.

5                                   b.       ECM HDLC Preamble Spoofing

10           The packet fax data exchange utilizes an HDLC frame structure for ECM high-speed data transmission. Preferably, the frame image data is divided by one or more HDLC preamble flags. If the network under-runs due to jitter or packet delay, the network gateways spoof their respective fax devices at the T.4 level by adding extra HDLC flags between frames. This spoofing technique increases the sending time to compensate for packet under-run due to network jitter and delay. Returning to FIG. 46 if the jitter buffer 1410 underflows, a buffer low indication 1410a is coupled to the spoofing logic 1416. Upon receipt of a buffer low indication during the fax data signal transmission, the spoofing logic 1416 inserts HDLC preamble flags at the frame boundary via the transmit fax data pump 1414. When the jitter buffer 1410 is filled to a pre-determined level, the fax image data is transferred out of the jitter buffer 1410.

20           In the described exemplary embodiment, the jitter buffer 1410 must be sized to store at least one HDLC frame so that a frame boundary may be located. The length of the largest T.4 ECM HDLC frame is 260 octets or 130 16-bit words. Spoofing is preferably activated when the number of packets stored in the jitter buffer 1410 drops to a predetermined threshold level. When spoofing is required, the spoofing logic 1416 adds HDLC flags at the frame boundary as a complete frame is being reassembled and forwarded to the transmit fax data pump 1414. This continues until the number of data packets in the jitter buffer 1410 exceeds the threshold level. The maximum time the network gateways will spoof their respective local fax devices can vary but can generally be about ten seconds.

25                                   c.       Non-ECM Spoofing with Zero Bit Filling

30           T.4 spoofing handles delay impairments during page transfer or C phase of a fax call. For those systems that do not utilize ECM, phase C signals comprise a series of coded image data followed by fill bits and end-of-line (EOL) sequences. Typically, fill bits are zeros inserted between the fax data signals and the EOL sequences, "000000000001". Fill bits ensure that a fax machine has time to perform the various mechanical overhead functions associated with any line it receives. Fill bits can also be utilized to spoof the jitter buffer to ensure compliance with the minimum transmission time of the total coded scan line established in the pre-message V.21 control procedure. The number of the bits of coded image contained in the data signals associated with the scan line and transmission speed limit the number of fill bits that can be added to the data signals. Preferably, the maximum transmission of any coded scan line is limited to less than about 5 sec. Thus, if the coded image for a given scan line contains 1000 bits

and the transmission rate is 2400 bps; then the maximum duration of fill time is  $(5 - (1000 + 12)/2400) = 4.57$  sec.

Generally, the packet fax data exchange utilizes spoofing if the network jitter delay exceeds the delay capability of the jitter buffer 1410. In accordance with the EOL spoofing method, fill bits can only be inserted immediately before an EOL sequence, so that the jitter buffer 1410 should preferably store at least one EOL sequence. Thus the jitter buffer 1410 should preferably be sized to hold at least one entire scan line of data to ensure the presence of at least one EOL sequence within the jitter buffer 1410. Thus, depending upon transmission rate, the size of the jitter buffer 1410 can become prohibitively large. The table below summarizes the desired jitter buffer data space to perform EOL spoofing for various scan line lengths. The table assumes that each pixel is represented by a single bit. The values represent an approximate upper limit on the required data space, but not the absolute upper limit, because in theory at least, the longest scan line can consist of alternating black and white pixels which would require an average of 4.5 bits to represent each pixel rather than the one to one ratio summarized in the table.

Scan Line Length	Number of words	sec to print out at 2400	sec to print out at 4800	sec to print out at 9600	sec to print out at 14400
1728	108	0.72	0.36	0.18	0.12
2048	128	0.853	0.427	0.213	0.14
2432	152	1.01	0.507	0.253	0.17
3456	216	1.44	0.72	0.36	0.24
4096	256	2	0.853	0.43	0.28
4864	304	2.375	1.013	0.51	0.34

To ensure the jitter buffer 1410 stores an EOL sequence, the spoofing logic 1416 should be activated when the number of data packets stored in the jitter buffer 1410 drops to a threshold level. Typically, a threshold value of about 200 msec is used to support the most commonly used fax setting, namely a fax speed of 9600 bps and scan line length of 1728. An alternate spoofing method should be used if an EOL sequence is not contained within the jitter buffer 1410, otherwise the call will have to be terminated. An alternate spoofing method uses zero run length code words. This method requires real time image data decoding so that the word boundary is known. Advantageously, this alternate method reduces the required size of the jitter buffer 1410.

Simply increasing the storage capacity of the jitter buffer 1410 can minimize the need for spoofing. However, overall network delay increases when the size of the jitter buffer 1410 is increased. Increased network delay may complicate the T.30 negotiation at the end of page or end of document, because of susceptibility to time out. Such a situation arises when the sending

1 fax machine completes the transmission of high speed data, and switches to an HDLC phase and  
sends the first V.21 packet in the end of page / multi-page signaling phase, (i.e. phase D). The  
sending fax machine must be kept alive until the response to the V.21 data packet is received.  
The receiving fax device requires more time to flush a large jitter buffer and then respond, hence  
5 complicating the T.30 negotiation.

In addition, the length of time a fax machine can be spoofed is limited, so that the jitter  
buffer 1410 can not be arbitrarily large. A pipeline store and forward relay is a combination of  
store and forward and spoofing techniques to approximate the performance of a typical Group  
3 fax connection when the network delay is large (on the order of seconds or more). One  
10 approach is to store and forward a single page at a time. However, this approach requires a  
significant amount of memory (10 Kwords or more). One approach to reduce the amount of  
memory required entails discarding scan lines on the sending network gateway and performing  
line repetition on the receiving network gateway so as to maintain image aspect ratio and quality.  
Alternatively, a partial page can be stored and forwarded thereby reducing the required amount  
15 of memory.

The sending and receiving fax machines will have some minimal differences in clock  
frequency. ITU standards recommends a data pump data rate of  $\pm 100$  ppm, so that the clock  
frequencies between the receiving and sending fax machines could differ by up to 200 ppm.  
Therefore, the data rate at the receiving network gateway (jitter buffer 1410) can build up or  
20 deplete at a rate of 1 word for every 5000 words received. Typically a fax page is less than 1000  
words so that end to end clock synchronization is not required.

### C. Data Relay Mode

25 Data relay mode provides full duplex signal processing of data signals. Data relay mode  
enables the transmission of data signals over a packet based system such as VoIP, VoFR, FRF-  
11, VTOA, or any other proprietary network. The data relay mode should also permit data  
signals to be carried over traditional media such as TDM. Network gateways, support the  
exchange of data signals other network gateways via a HFC network and CMTS or off line  
30 devices via for example a circuit switched network such as the PSTN. For the purposes of  
explanation, the first modem is referred to as a call modem. Far end modems are typically called  
answer modems.

In data relay mode, a local modem connection is established on each end of the packet  
based network 1494. That is, the call modem 1492a and the call network gateway 496a establish  
35 a local modem connection, as does the destination answer modem 1492b and its respective  
answer network gateway 1496b. Next, data signals are relayed across the packet based network  
1494. The call network gateway 1496a demodulates the data signal and formats the demodulated  
data signal for the particular packet based network 1494. The answer network gateway 1496b  
compensates for network impairments and remodulates the encoded data in a format suitable for

1 the destination answer modem 1492b. This approach results in considerable bandwidth savings since only the underlying demodulated data signals are transmitted across the packet based network.

5 In the data relay mode, the packet data modem exchange provides demodulation and modulation of data signals. With full duplex capability, both modulation and demodulation of data signals can be performed simultaneously. The packet data modem exchange also provides compensation for network jitter with a jitter buffer similar to that invoked in the packet voice exchange. Additionally, the packet data modem exchange compensates for system clock jitter between modems with a dynamic phase adjustment and resampling mechanism. Spoofing may  
10 also be provided during various stages of the call negotiation procedure between the modems to keep the connection alive.

The packet data modem exchange invoked by the network VHD in the data relay mode is shown schematically in FIG. 49. In the described exemplary embodiment, a connecting PXD (not shown) connecting a modem to the switch board 1032' is transparent, although those skilled  
15 in the art will appreciate that various signal conditioning algorithms could be programmed into the PXD such as filtering, echo cancellation and gain.

After the PXD, the data signals are coupled to the network VHD via the switchboard 1032'. The packet data modem exchange provides two way communication between a circuit  
20 switched network and packet based network with two basic functional units, a demodulation system and a remodulation system. In the demodulation system, the network VHD exchanges data signals from a circuit switched network, or a telephony device directly, to a packet based network. In the remodulation system, the network VHD exchanges data signals from the packet based network to the PSTN line, or the telephony device.

25 In the demodulation system, the data signals are received and buffered in an ingress media queue 1500. A data pump receiver 1504 demodulates the data signals from the ingress media queue 1500. The data pump receiver 1504 supports the V.22bis standard for the demodulation of data signals at 1200/2400 bps; the V.32bis standard for the demodulation of data signals at 4800/7200/9600/12000/14400 bps, as well as the V.34 standard for the demodulation of data  
30 signals up to 33600 bps. Moreover, the V.90 standard may also be supported. The demodulated data signals are then packetized by the packetization engine 1506 and transmitted across the packet based network.

In the remodulation system, packets of data signals from the packet based network are first depacketized by a depacketizing engine 1508 and stored in a jitter buffer 1510. A data pump  
35 transmitter 1512 modulates the buffered data signals with a voiceband carrier. The modulated data signals are in turn stored in the egress media queue 1514 before being output to the PXD (not shown) via the switchboard 1032'. The data pump transmitter 1512 supports the V.22bis standard for the transfer of data signals at 1200/2400 bps; the V.32bis standard for the transfer

1 of data signals at 4800/7200/9600/12000/14400 bps. as well as the V.34 standard for the transfer of data signal up to 33600 bps. Moreover, the V.90 standard may also be supported.

5 During jitter buffer underflow, the jitter buffer 1510 sends a buffer low indication 1510a to spoofing logic 1516. When the spoofing logic 1516 receives the buffer low signal indicating that the jitter buffer 1510 is operating below a predetermined threshold level, it inserts spoofed data at the appropriate place in the data signal via the data pump transmitter 1512. Spoofing continues until the jitter buffer 1510 is filled to the predetermined threshold level, at which time data signals are again transferred from the jitter buffer 1510 to the data pump transmitter 1512.

10 End to end clock logic 1518 also monitors the state of the jitter buffer 1510. The clock logic 1518 controls the data transmission rate of the data pump transmitter 1512 in correspondence to the state of the jitter buffer 1510. When the jitter buffer 1510 is below a predetermined threshold level, the clock logic 1518 reduces the transmission rate of the data pump transmitter 1512. Likewise, when the jitter buffer 1510 is above a predetermined threshold level, the clock logic 1518 increases the transmission rate of the data pump transmitter 1512.

15 Before the transmission of data signals across the packet based network, the connection between the two modems must first be negotiated through a handshaking sequence. This entails a two-step process. First, a call negotiator 1502 determines the type of modem (i.e., V.22, V.32bis, V.34, V.90, etc.) connected to each end of the packet based network. Second, a rate negotiator 1520 negotiates the data signal transmission rate between the two modems.

20 The call negotiator 1502 determines the type of modem connected locally, as well as the type of modem connected remotely via the packet based network. The call negotiator 1502 utilizes V.25 automatic answering procedures and V.8 auto-baud software to automatically detect modem capability. The call negotiator 1502 receives protocol indication signals 1502a (ANSam and V.8 menus) from the ingress media queue 1500, as well as AA, AC and other message indications 1502b from the local modem via a data pump state machine 1522, to determine the type of modem in use locally. The call negotiator 1502 relays the ANSam answer tones and other indications 1502e from the data pump state machine 1522 to the remote modem via a packetization engine 1506. The call negotiator also receives ANSam, AA, AC and other indications 1502c from a remote modem (not shown) located on the opposite end of the packet based network via a depacketizing engine 1508. The call negotiator 1502 relays ANSam answer tones and other indications 1502d to a local modem (not shown) via an egress media queue 1514 of the modulation system. With the ANSam, AA, AC and other indications from the local and remote modems, the call negotiator 1502 can then negotiate a common standard (i.e., V.22, V.32bis, V.34, V.90, etc.) in which the data pumps must communicate with the local modem and the remote modems.

30 The packet data modem exchange preferably utilizes indication packets as a means for communicating answer tones, AA, AC and other indication signals across the packet based



1 network. However, the packet data modem exchange supports data pumps such as V.22bis and  
V.32bis which do not include a well defined error recovery mechanism, so that the modem  
connection may be terminated whenever indication packets are lost. Therefore, either the packet  
data modem exchange or the application layer should ensure proper delivery of indication packets  
5 when operating in a network environment that does not guarantee packet delivery.

The packet data modem exchange can ensure delivery of the indication packets by  
periodically retransmitting the indication packet until some expected packets are received. For  
example, in V.32bis relay, the call negotiator operating under the packet data modem exchange  
on the answer network gateway periodically retransmits ANSam answer tones from the answer  
10 modem to the call modem, until the calling modem connects to the line and transmits carrier  
state AA.

Alternatively, the packetization engine can embed the indication information directly into  
the packet header. In this approach, an alternate packet format is utilized to include the  
indication information. During modem handshaking, indication packets transmitted across the  
15 packet based network include the indication information, so that the system does not rely on the  
successful transmission of individual indication packets. Rather, if a given packet is lost, the  
next arriving packet contains the indication information in the packet header. Both methods  
increase the traffic across the network. However, it is preferable to periodically retransmit the  
indication packets because it has less of a detrimental impact on network traffic.

20 A rate negotiator 1520 synchronizes the connection rates at the network gateways 1496a,  
1496b, 1496c (see FIG. 49). The rate negotiator receives rate control codes 1520a from the local  
modem via the data pump state machine 1522 and rate control codes 1520b from the remote  
modem via the depacketizing engine 1508. The rate negotiator 1520 also forwards the remote  
rate control codes 1520a received from the remote modem to the local modem via commands  
25 sent to the data pump state machine 1522. The rate negotiator 1520 forwards the local rate control  
codes 1520c received from the local modem to the remote modem via the packetization engine  
1506. Based on the exchanged rate codes the rate negotiator 1520 establishes a common data rate  
between the calling and answering modems. During the data rate exchange procedure, the jitter  
buffer 1510 should be disabled by the rate negotiator 1520 to prevent data transmission between  
30 the call and answer modems until the data rates are successfully negotiated.

Similarly error control (V.42) and data compression (V.42bis) modes should be  
synchronized at each end of the packet based network. Error control logic 1524 receives local  
error control messages 1524a from the data pump receiver 1504 and forwards those V.14/V.42  
negotiation messages 1524c to the remote modem via the packetization engine 1506. In addition,  
35 error control logic 1524 receives remote V.14/V.42 indications 1524b from the depacketizing  
engine 1508 and forwards those V.14/V.42 indications 1524d to the local modem. With the  
V.14/V.42 indications from the local and remote modems, the error control logic 1524 can  
negotiate a common standard to ensure that the network gateways utilize a common error

1 protocol. In addition, error control logic 1524, communicates the negotiated error control  
protocol 1524(e) to the spoofing logic 1516 to ensure data mode spoofing is in accordance with  
the negotiated error control mode.

5 V.42 is a standard error correction technique using advanced cyclical redundancy checks  
and the principle of automatic repeat requests (ARQ). In accordance with the V.42 standard,  
transmitted data signals are grouped into blocks and cyclical redundancy calculations add error  
checking words to the transmitted data signal stream. The receiving modem calculates new error  
check information for the data signal block and compares the calculated information to the  
10 received error check information. If the codes match, the received data signals are valid and  
another transfer takes place. If the codes do not match, a transmission error has occurred and the  
receiving modem requests a repeat of the last data block. This repeat cycle continues until the  
entire data block has been received without error.

Various voiceband data modem standards exist for error correction and data compression.  
15 V.42bis and MNP5 are examples of data compression standards. The handshaking sequence for  
every modem standard is different so that the packet data modem exchange should support  
numerous data transmission standards as well as numerous error correction and data compression  
techniques.

#### 20 1. End to End Clock Logic

Slight differences in the clock frequency of the call modem and the answer modem are  
expected, since the baud rate tolerance for a typical modem data pump is  $\pm 100$  ppm. This  
tolerance corresponds to a relatively low depletion or build up rate of 1 in 5000 words. However,  
the length of a modem session can be very long, so that uncorrected difference in clock frequency  
25 may result in jitter buffer underflow or overflow.

In the described exemplary embodiment, the clock logic synchronizes the transmit clock  
of the data pump transmitter 1512 to the average rate at which data packets arrive at the jitter  
buffer 1510. The data pump transmitter 1512 packages the data signals from the jitter buffer  
1510 in frames of data signals for demodulation and transmission to the egress media queue  
30 1514. At the beginning of each frame of data signals, the data pump transmitter 1512 examines  
the egress media queue 1514 to determine the remaining buffer space, and in accordance  
therewith, the data pump transmitter 1512 modulates that number of digital data samples required  
to produce a total of slightly more or slightly less than 80 samples per frame, assuming that the  
data pump transmitter 1512 is invoked once every 10 msec. The data pump transmitter 1512  
35 gradually adjusts the number of samples per frame to allow the receiving modem to adjust to the  
timing change. Typically, the data pump transmitter 1512 uses an adjustment rate of about one  
ppm per frame. The maximum adjustment should be less than about 200 ppm.

1 In the described exemplary embodiment, end to end clock logic 1518 monitors the space  
available within the jitter buffer 1510 and utilizes water marks to determine whether the data rate  
of the data pump transmitter 1512 should be adjusted. Network jitter may cause timing  
adjustments to be made. However, this should not adversely affect the data pump receiver of the  
5 answering modem as these timing adjustments are made very gradually.

## 2. Modem Connection Handshaking Sequence.

### a. Call Negotiation.

10 A single industry standard for the transmission of modem data over a packet based  
network does not exist. However, numerous common standards exist for transmission of modem  
data at various data rates over the PSTN. For example, V.22 is a common standard used to  
define operation of 1200 bps modems. Data rates as high as 2400 bps can be implemented with  
the V.22bis standard (the suffix "bis" indicates that the standard is an adaptation of an existing  
15 standard). The V.22bis standard groups data signals into four bit words which are transmitted  
at 600 baud. The V.32 standard supports full duplex, data rates of up to 9600 bps over the PSTN.  
A V.32 modem groups data signals into four bit words and transmits at 2400 baud. The V.32bis  
standard supports duplex modems operating at data rates up to 14,400 bps on the PSTN. In  
addition, the V.34 standard supports data rates up to 33,600 bps on the public switched telephone  
20 network. In the described exemplary embodiment, these standards can be used for data signal  
transmission over the packet based network with a call negotiator that supports each standard.

### b. Rate Negotiation.

25 Rate negotiation refers to the process by which two telephony devices are connected at  
the same data rate prior to data transmission. In the context of a modem connection in  
accordance with an exemplary embodiment of the present invention, each modem is coupled to  
a signal processing system, which for the purposes of explanation is operating in a network  
gateway, either directly or through a PSTN line. In operation, each modem establishes a modem  
connection with its respective network gateway, at which point, the modems begin relaying data  
30 signals across a packet based network. The problem that arises is that each modem may negotiate  
a different data rate with its respective network gateway, depending on the line conditions and  
user settings. In this instance, the data signals transmitted from one of the modems will enter the  
packet based network faster than it can be extracted at the other end by the other modem. The  
resulting overflow of data signals may result in a lost connection between the two modems. To  
prevent data signal overflow, it is, therefore, desirable to ensure that both modems negotiate to  
35 the same data rate. A rate negotiator can be used for this purpose. Although the the rate  
negotiator is described in the context of a signal processing system with the packet data modem  
exchange invoked, those skilled in the art will appreciate that the rate negotiator is likewise  
suitable for various other telephony and telecommunications application. Accordingly, the

1 described exemplary embodiment of the rate negotiator in a signal processing system is by way of example only and not by way of limitation.

5 In an exemplary embodiment, data rate negotiation is achieved through a data rate negotiation procedure, wherein a call modem independently negotiates a data rate with a call network gateway, and an answer modem independently negotiates a data rate with an answer network gateway. The calling and answer network gateways, each having a signal processing system running a packet exchange, then exchange data packets containing information on the independently negotiated data rates. If the independently negotiated data rates are the same, then each rate negotiator will enable its respective network gateway and data transmission between the call and answer modems will commence. Conversely, if the independently negotiated data rates are different, the rate negotiator will renegotiate the data rate by adopting the lowest of the two data rates. The call and answer modems will then undergo retraining or rate renegotiation procedures by their respective network gateways to establish a new connection at the renegotiated data rate. The advantage of this approach is that the data rate negotiation procedure takes advantage of existing modem functionality, namely, the retraining and rate renegotiation mechanism, and puts it to alternative usage. Moreover, by retraining both the call and answer modem (one modem will already be set to the renegotiated rate) both modems are automatically prevented from sending data.

20 Alternatively, the calling and answer modems can directly negotiate the data rate. This method is not preferred for modems with time constrained handshaking sequences such as, for example, modems operating in accordance with the V.22bis or the V.32bis standards. The round trip delay accommodated by these standards could cause the modem connection to be lost due to timeout. Instead, retrain or rate renegotiation should be used for data signals transferred in accordance with the V.22bis and V.32bis standards, whereas direct negotiation of the data rate by the local and remote modems can be used for data exchange in accordance with the V.34 and V.90 (a digital modem and analog modem pair for use on PSTN lines at data rates up to 56,000 bps downstream and 33,600 upstream) standards.

c. Exemplary Handshaking Sequences.

30 (V.22 Handshaking Sequence)

35 The call negotiator on the answer network gateway, differentiates between modem types and relays the ANSam answer tone. The answer modem transmits unscrambled binary ones signal (USB1) indications to the answer mode gateway. The answer network gateway forwards USB1 signal indications to the call network gateway. The call negotiator in the call network gateway assumes operation in accordance with the V.22bis standard as a result of the USB1 signal indication and terminates the call negotiator. The packet data modem exchange, in the answer network gateway then invokes operation in accordance with the V.22bis standard after an answer tone timeout period and terminates its call negotiator.

1 V.22bis handshaking does not utilize rate messages or signaling to indicate the selected  
bit rate as with most high data rate pumps. Rather, the inclusion of a fixed duration signal (S1)  
indicates that 2400 bps operation is to be used. The absence of the S1 signal indicates that 1200  
bps should be selected. The duration of the S1 signal is typically about 100 msec, making it  
likely that the call modem will perform rate determination (assuming that it selects 2400 bps)  
5 before rate indication from the answer modem arrives. Therefore, the rate negotiator in the call  
network gateway should select 2400 bps operation and proceed with the handshaking procedure.  
If the answer modem is limited to a 1200 bps connection, rate renegotiation is typically used to  
change the operational data rate of the call modem to 1200 bps. Alternatively, if the call modem  
selects 1200 bps, rate renegotiation would not be required.

#### 10 (V.32bis Handshaking Sequence)

V32bis handshaking utilizes rate signals (messages) to specify the bit rate. A relay  
sequence in accordance with the V.32bis standard is shown in FIG. 50 and begins with the call  
negotiator in the answer network gateway relaying ANSam 1530 answer tone from the answer  
15 modem to the call modem. After receiving the answer tone for a period of at least one second,  
the call modem connects to the line and repetitively transmits carrier state A 1532. When the call  
network gateway detects the repeated transmission of carrier state A ("AA"), the call network  
gateway relays this information 1534 to the answer network gateway. In response the answer  
network gateway forwards the AA indication to the answer modem and invokes operation in  
20 accordance with the V.32bis standard. The answer modem then transmits alternating carrier  
states A and C 1536 to the answer network gateway. If the answer network gateway receives AC  
from the answer modem, the answer network gateway relays AC 1538 to the call network  
gateway, thereby establishing operation in accordance with the V.32bis standard, allowing call  
negotiator in the call network gateway to be terminated. Next, data rate alignment is achieved  
25 by either of two methods.

In the first method for data rate alignment of a V.32bis relay connection, the call modem  
and the answer modem independently negotiate a data rate with their respective network  
gateways at each end of the network 1540 and 1542. Next, each network gateway forwards a  
connection data rate indication 1544 and 1546 to the other network gateway. Each network  
30 gateway compares the far end data rate to its own data rate. The preferred rate is the minimum  
of the two rates. Rate renegotiation 1548 and 1550 is invoked if the connection rate of either  
network gateway to its respective modem differs from the preferred rate.

In the second method, rate signals R1, R2 and R3, are relayed to achieve data rate  
negotiation. FIG. 51 shows a relay sequence in accordance with the V.32bis standard for this  
35 alternate method of rate negotiation. The call negotiator relays the answer tone (ANSam) 1552  
from the answer modem to the call modem. When the call modem detects answer tone, it  
repetitively transmits carrier state A 1554 to the call network gateway. The call network gateway  
relays this information (AA) 1556 to the answer network gateway. The answer network gateway

1 sends the AA 1558 to the answer modem and initiates normal range tone exchange with the answer modem. The answer network gateway then forwards AC 1560 to call network gateway which in turn relays this information 1562 to the call modem to initiate normal range tone exchange between the call network gateway and the call modem.

5 The answer modem sends its first training sequence 1564 followed by R1 (the data rates currently available in the answer modem) to the rate negotiator in the answer network gateway. When the answer network gateway receives an R1 indication, it forwards R1 1566 to the call network gateway. The answer network gateway then repetitively sends training sequences to the answer modem. The call network gateway forwards the R1 indication 1570 of the answer  
10 modem to the call modem. The call modem sends training sequences to the call network gateway 1572. The call network gateway determines the data rate capability of the call modem, and forwards the data rate capabilities of the call modem to the answer network gateway in a data rate signal format. The call modem also sends an R2 indication 1568 (data rate capability of the call  
15 modem, preferably excluding rates not included in the previously received R1 signal, i.e. not supported by the answer modem) to the call network gateway which forwards it to the answer network gateway. The call network gateway then repetitively sends training sequences to the call modem until receiving an R3 signal 1574 from the answer modem via the answer network gateway.

20 The answer network gateway performs a logical AND operation on the R1 signal from the answer modem (data rate capability of the answer modem), the R2 signal from the call modem (data rate capability of the call modem, excluding rates not supported by the answer modem) and the training sequences of the call network gateway (data rate capability of the call modem) to create a second rate signal R2 1576, which is forwarded to the answer modem. The answer modem sends its second training sequence followed an R3 signal, which indicates the  
25 data rate to be used by both modems. The answer network gateway relays R3 1574 to the call network gateway which forwards it to the call modem and begins operating at the R3 specified bit rate. However, this method of rate synchronization is not preferred for V.32bis due to time constrained handshaking.

#### (V.34 Handshaking Sequence)

30 Data transmission in accordance with the V.34 standard utilizes a modulation parameter (MP) sequence to exchange information pertaining to data rate capability. The MP sequences can be exchanged end to end to achieve data rate synchronization. Initially, the call negotiator in the answer network gateway relays the answer tone (ANSam) from the answer modem to the call modem. When the call modem receives answer tone, it generates a CM indication and  
35 forwards it to the call network gateway. When the call network gateway receives a CM indication, it forwards it to the answer network gateway which then communicates the CM indication with the answer modem. The answer modem then responds by transmitting a JM sequence to the answer network gateway, which is relayed by the answer network gateway to the

1 call modem via the call network gateway. If the call network gateway then receives a CJ sequence  
from the call modem, the call negotiator in the call network gateway, initiates operation in  
accordance with the V.34 standard, and forwards a CJ sequence to the answer network gateway.  
If the JM menu calls for V.34, the call negotiator in the answer network gateway initiates  
5 operation in accordance with the V.34 standard and the call negotiator is terminated. If a  
standard other than V.34 is called for, the appropriate procedure is invoked, such as those  
described previously for V.22 or V.32bis. Next, data rate alignment is achieved by either of two  
methods.

10 In a first method for data rate alignment after a V.34 relay connection is established, the  
call modem and the answer modem freely negotiate a data rate at each end of the network with  
their respective network gateways. Each network gateway forwards a connection rate indication  
to the other gateway. Each gateway compares the far end bit rate to the rate transmitted by each  
gateway. For example, the call network gateway compares the data rate indication received from  
the answer modem gateway to that which it negotiated freely negotiated to with the call modem.  
15 The preferred rate is the minimum of the two rates. Rate renegotiation is invoked if the  
connection rate at the calling or receiving end differs from the preferred rate, to force the  
connection to the desired rate.

20 In an alternate method for V.34 rate synchronization, MP sequences are utilized to  
achieve rate synchronization without rate renegotiation. The call modem and the answer modem  
independently negotiate with the call network gateway and the answer network gateway  
respectively until phase IV of the negotiations is reached. The call network gateway and the  
answer network gateway exchange training results in the form of MP sequences when Phase IV  
of the independent negotiations is reached to establish the primary and auxiliary data rates. The  
call network gateway and the answer network gateway are preferably prevented from relaying MP  
sequences to the call modem and the answer modem respectively until the training results for  
25 both network gateways and the MP sequences for both modems are available. If symmetric rate  
is enforced, the maximum answer data rate and the maximum call data rate of the four MP  
sequences are compared. The lower data rate of the two maximum rates is the preferred data rate.  
Each network gateway sends the MP sequence with the preferred rate to its respective modem  
so that the calling and answer modems operate at the preferred data rate.

30 If asymmetric rates are supported, then the preferred call-answer data rate is the lesser  
of the two highest call-answer rates of the four MP sequences. Similarly, the preferred answer-  
call data rate is the lesser of the two highest answer-call rates of the four MP sequences. Data  
rate capabilities may also need to be modified when the MP sequence are formed so as to be sent  
to the calling and answer modems. The MP sequence sent to the calling and answer modems,  
35 is the logical AND of the data rate capabilities from the four MP sequences.

(V.90 Handshaking Sequence)

1       The V.90 standard utilizes a digital and analog modem pair to transmit modem data over  
the PSTN line. The V.90 standard utilizes MP sequences to convey training results from a digital  
to an analog modem, and a similar sequence, using constellation parameters (CP) to convey  
training results from an analog to a digital modem. Under the V.90 standard, the timeout period  
5       is 15 seconds compared to a timeout period of 30 seconds under the V.34 standard. In addition,  
the analog modems control the handshake timing during training. In an exemplary embodiment,  
the call modem and the answer modem are the V.90 analog modems. As such the call modem  
and the answer modem are beyond the control of the network gateways during training. The  
digital modems only control the timing during transmission of TRN1d, which the digital modem  
10       in the network gateway uses to train its echo canceller.

      When operating in accordance with the V.90 standard, the call negotiator utilizes the V.8  
recommendations for initial negotiation. Thus, the initial negotiation of the V.90 relay session  
is substantially the same as the relay sequence described for V.34 rate synchronization method  
one and method two with asymmetric rate operation. There are two configurations where V.90  
15       relay may be used. The first configuration is data relay between two V.90 analog modems, i.e.  
each of the network gateways are configured as V.90 digital modems. The upstream rate between  
two V.90 analog modems, according to the V.90 standard, is limited to 33,600 bps. Thus, the  
maximum data rate for an analog to analog relay is 33,600 bps. In accordance with the V.90  
standard, the minimum data rate a V.90 digital modem will support is 28,800 bps. Therefore,  
the connection must be terminated if the maximum data rate for one or both of the upstream  
20       directions is less than 28,800 bps, and one or both the downstream direction is in V.90 digital  
mode. Therefore, the V.34 protocol is preferred over V.90 for data transmission between local  
and remote analog modems.

      A second configuration is a connection between a V.90 analog modem and a V.90 digital  
25       modem. A typical example of such a configuration is when a user within a packet based PABX  
system dials out into a remote access server (RAS) or an Internet service provider (ISP) that uses  
a central site modem for physical access that is V.90 capable. The connection from PABX to the  
central site modem may be either through PSTN or directly through an ISDN, T1 or E1 interface.  
Thus the V.90 embodiment should preferably support an analog modem interfacing directly to  
ISDN, T1 or E1.

30       For an analog to digital modem connection, the connections at both ends of the packet  
based network should be either digital or analog to achieve proper rate synchronization. The  
analog modem decides whether to select digital mode as specified in INFO1a, so that INFO1a  
should be relayed between the calling and answer modem via their respective network gateways  
before operation mode is synchronized.

35       Upon receipt of an INFO1a signal from the answer modem, the answer network gateway  
performs a line probe on the signal received from the answer modem to determine whether digital  
mode can be used. The call network gateway receives an INFO1a signal from the call modem.



1 The call network gateway sends a mode indication to the answer network gateway indicating whether digital or analog will be used and initiates operation in the mode specified in INFO1a. Upon receipt of an analog mode indication signal from the call network gateway, the answer network gateway sends an INFO1a sequence to the answer modem. The answer network gateway then proceeds with analog mode operation. Similarly, if digital mode is indicated and digital mode can be supported by the answer modem, the answer network gateway sends an INFO1a sequence to the answer modem indicating that digital mode is desired and proceeds with digital mode operation.

10 Alternatively, if digital mode is indicated and digital mode can not be supported by the answer modem, the call modem should preferably be forced into analog mode by one of three alternate methods. First, some commercially available V.90 analog modems may revert to analog mode after several retrains. Thus, one method to force the call modem into analog mode is to force retrains until the call modem selects analog mode operation. In an alternate method, the call network gateway modifies its line probe so as to force the call modem to select analog mode. In a third method, the call modem and the answer modem operate in different modes. Under this method if the answer modem can not support a 28,800 bps data rate the connection is terminated.

### 3. Data Mode Spoofing

20 The jitter buffer 1510 may underflow during long delays of data signal packets. Jitter buffer underflow can cause the data pump transmitter 1512 to run out of data, and therefore, it is desirable that the jitter buffer 1510 be spoofed with bit sequences. Preferably the bit sequences are benign. In the described exemplary embodiment, the specific spoofing methodology is dependent upon the common error mode protocol negotiated by the error control logic of each network gateway.

25 In accordance with V.14 recommendations, the spoofing logic 1516 checks for character format and boundary (number of data bits, start bits and stop bits) within the jitter buffer 1510. As specified in the V.14 recommendation the spoofing logic 1516 must account for stop bits omitted due to asynchronous-to-synchronous conversion. Once the spoofing logic 1516 locates the character boundary, ones can be added to spoof the local modem and keep the connection alive. The length of time a modem can be spoofed with ones depends only upon the application program driving the local modem.

30 In accordance with the V.42 recommendations, the spoofing logic 1516 checks for HDLC flag (HDLC frame boundary) within the jitter buffer 1510. The basic HDLC structure consists of a number of frames, each of which is subdivided into a number of fields. The HDLC frame structure provides for frame labeling and error checking. When a new data transmission is initiated, HDLC preamble in the form of synchronization sequences are transmitted prior to the binary coded information. The HDLC preamble is modulated bit streams of "01111110

1 (0x7e)". The jitter buffer 1510 should be sufficiently large to guarantee that at least one complete  
HDLC frame is contained within the jitter buffer 1510. The default length of an HDLC frame  
is 132 octets. The V.42 recommendations for error correction of data circuit terminating  
5 equipment (DCE) using asynchronous-to-synchronous conversion does not specify a maximum  
length for an HDLC frame. However, because the length of the frame affects the overall  
memory required to implement the protocol, a information frame length larger than 260 octets  
is unlikely.

10 The spoofing logic 1516 stores a threshold water mark (with a value set to be  
approximately equal to the maximum length of the HDLC frame). Spoofing is preferably  
activated when the number of packets stored in the jitter buffer 1510 drops to the predetermined  
threshold level. When spoofing is required, the spoofing logic 1516 adds HDLC flags at the  
frame boundary as a complete frame is being reassembled and forwarded to the transmit data  
pump. This continues until the number of data packets in the jitter buffer 1510 exceeds the  
threshold level.

#### 15 4. Retrain and Rate Renegotiation

In the described exemplary embodiment, if data rates independently negotiated between  
the modems and their respective network gateways are different, the rate negotiator will  
renegotiate the data rate by adopting the lowest of the two data rates. The call and answer  
20 modems will then undergo retraining or rate renegotiation procedures by their respective network  
gateways to establish a new connection at the renegotiated data rate. In addition, rate  
synchronization may be lost during a modem communication, requiring modem retraining and  
rate renegotiation, due to drift or change in the conditions of the communication channel. When  
a retrain occurs, an indication should be forwarded to the network gateway at the end of the  
25 packet based network. The network gateway receiving a retrain indication should initiate retrain  
with the connected modem to keep data flow in synchronism between the two connections. Rate  
synchronization procedures as previously described should be used to maintain data rate  
alignment after retrains.

30 Similarly, rate renegotiation causes both the calling and answer network gateways and  
to perform rate renegotiation. However, rate signals or MP (CP) sequences should be exchanged  
per method two of the data rate alignment as previously discussed for a V.32bis or V.34 rate  
synchronization whichever is appropriate.

#### 35 5. Error Correcting Mode Synchronization

Error control (V.42) and data compression (V.42bis) modes should be synchronized at  
each end of the packet based network. In a first method, the call modem and the answer modem  
independently negotiate an error correction mode with each other on their own. transparent to the

1 network gateways. This method is preferred for connections wherein the network delay plus jitter is relatively small, as characterized by an overall round trip delay of less than 700 msec.

5 Data compression mode is negotiated within V.42 so that the appropriate mode indication can be relayed when the calling and answer modems have entered into V.42 mode. An alternative method is to allow modems at both ends to freely negotiate the error control mode with their respective network gateways. The network gateways must fully support all error correction modes when using this method. Also, this method cannot support the scenario where one modem selects V.14 while the other modem selects a mode other than V.14. For the case where V.14 is negotiated at both sides of the packet based network, an 8-bit no parity format is assumed by each respective network gateway and the raw demodulated data bits are transported there between. With all other cases, each gateway shall extract de-framed (error corrected) data bits and forward them to its counterpart at the opposite end of the network. Flow control procedures within the error control protocol may be used to handle network delay. The advantage of this method over the first method is its ability to handle large network delays and also the scenario where the local connection rates at the network gateways are different. However, packets transported over the network in accordance with this method must be guaranteed to be error free. This may be achieved by establishing a connection between the network gateways in accordance with the link access protocol connection for modems (LAPM)

## 6. Data Pump

20 Preferably, the data exchange includes a modem relay having a data pump for demodulating modem data signals from a modem for transmission on the packet based network, and remodulating modem data signal packets from the packet based network for transmission to a local modem. Similarly, the data exchange also preferably includes a fax relay with a data pump for demodulating fax data signals from a fax for transmission on the packet based network, and remodulating fax data signal packets from the packet based network for transmission to a local fax device. The utilization of a data pump in the fax and modem relays to demodulate and remodulate data signals for transmission across a packet based network provides considerable bandwidth savings. First, only the underlying unmodulated data signals are transmitted across the packet based network. Second, data transmission rates of digital signals across the packet based network, typically 64 kbps is greater than the maximum rate available (typically 33,600 bps) for communication over a circuit switched network.

35 Telephone line data pumps operating in accordance with ITU V series recommendations for transmission rates of 2400 bps or more typically utilize quadrature amplitude modulation (QAM). A typical QAM data pump transmitter 1600 is shown schematically in FIG. 52. The transmitter input is a serial binary data stream  $d_n$  arriving at a rate of  $R_d$  bps. A serial to parallel converter 1602 groups the input bits into J-bit binary words. A constellation mapper 1604 maps each J-bit binary word to a channel symbol from a  $2^J$  element alphabet resulting in a channel symbol rate of  $f_s = R_d/J$  baud. The alphabet consists of a pair of real

1 numbers representing points in a two-dimensional space, called the signal constellation. Customarily the signal constellation can be thought of as a complex plane so that the channel symbol sequence may be represented as a sequence of complex numbers  $c_n = a_n + jb_n$ . Typically the real part  $a_n$  is called the in-phase or I component and the imaginary  $b_n$  is called the quadrature or Q component. A nonlinear encoder 1605 may be used to expand the constellation points in order to combat the negative effects of companding in accordance with ITU-T G.711 standard. The I & Q components may be modulated by impulse modulators 1606 and 1608 respectively and filtered by transmit shaping filters 1610 and 1612 each with impulse response  $g_T(t)$ . The outputs of the shaping filters 1610 and 1612 are called in-phase 1610(a) and quadrature 1612(a) components of the continuous-time transmitted signal.

10 The shaping filters 1610 and 1612 are typically lowpass filters approximating the raised cosine or square root of raised cosine response, having a cutoff frequency on the order of at least about  $f_c/2$ . The outputs 1610(a) and 1612(a) of the lowpass filters 1610 and 1612 respectively are lowpass signals with a frequency domain extending down to approximately zero hertz. A local oscillator 1614 generates quadrature carriers  $\cos(\omega_c t)$  1614(a) and  $\sin(\omega_c t)$  1614(b). Multipliers 1616 and 1618 multiply the filter outputs 1610(a) and 1612(a) by quadrature carriers  $\cos(\omega_c t)$  and  $\sin(\omega_c t)$  respectively to amplitude modulate the in-phase and quadrature signals up to the passband of a bandpass channel. The modulated output signals 1616(a) and 1618(a) are then subtracted in a difference operator 1620 to form a transmit output signal 1622. The carrier frequency should be greater than the shaping filter cutoff frequency to prevent spectral fold-over.

20 A data pump receiver 1630 is shown schematically in FIG. 53. The data pump receiver 1630 is generally configured to process a received signal 1630(a) distorted by the non-ideal frequency response of the channel and additive noise in a transmit data pump (not shown) in the local modem. An analog to digital converter (A/D) 1631 converts the received signal 1630(a) from an analog to a digital format. The A/D converter 1631 samples the received signal 1630(a) at a rate of  $f_s = 1/T_s = n_s/T$  which is  $n_s$  times the symbol rate  $f_s = 1/T$  and is at least twice the highest frequency component of the received signal 1630(a) to satisfy nyquist sampling theory.

30 An echo canceller 1634 substantially removes the line echos on the received signal 1630(a). Echo cancellation permits a modem to operate in a full duplex transmission mode on a two-line circuit, such as a PSTN. With echo cancellation, a modem can establish two high-speed channels in opposite directions. Through the use of digital-signal-processing circuitry, the modem's receiver can use the shape of the modem's transmitter signal to cancel out the effect of its own transmitted signal by subtracting reference signal and the receive signal 1630(a) in a difference operator 1633.

35 Multiplier 1636 scales the amplitude of echo cancelled signal 1633(a). A power estimator 1637 estimates the power level of the gain adjusted signal 1636(a). Automatic gain control logic 1638 compares the estimated power level to a set of predetermined thresholds and

1 inputs a scaling factor into the multiplier 1636 that adjusts the amplitude of the echo canceled  
signal 1633(a) to a level that is within the desired amplitude range. A carrier detector 1642  
processes the output of a digital resampler 1640 to determine when a data signal is actually  
present at the input to receiver 1630. Many of the receiver functions are preferably not invoked  
5 until an input signal is detected.

A timing recovery system 1644 synchronizes the transmit clock of the remote data pump  
transmitter (not shown) and the receiver clock. The timing recovery system 1644 extracts timing  
information from the received signal, and adjusts the digital resampler 1640 to ensure that the  
frequency and phase of the transmit clock and receiver clock are synchronized. A phase splitting  
10 fractionally spaced equalizer (PSFSE) 1646 filters the received signal at the symbol rate. The  
PSFSE 1646 compensates for the amplitude response and envelope delay of the channel so as to  
minimize inter-symbol interference in the received signal. The frequency response of a typical  
channel is inexact so that an adaptive filter is preferable. The PSFSE 1646 is preferably an  
adaptive FIR filter that operates on data signal samples spaced by  $T/n_0$  and generates digital  
15 signal output samples spaced by the period  $T$ . In the described exemplary embodiment  $n_0=3$ .

The PSFSE 1646 outputs a complex signal which multiplier 1650 multiplies by a locally  
generated carrier reference 1652 to demodulate the PSFSE output to the baseband signal 1650(a).  
The received signal 1630(a) is typically encoded with a non-linear operation so as to reduce the  
quantization noise introduced by companding in accordance with ITU-T G.711. The baseband  
20 signal 1650(a) is therefore processed by a non-linear decoder 1654 which reverses the non-linear  
encoding or warping. The gain of the baseband signal will typically vary upon transition from  
a training phase to a data phase because modem manufacturers utilize different methods to  
compute a scale factor. The problem that arises is that digital modulation techniques such as  
quadrature amplitude modulation (QAM) and pulse amplitude modulation (PAM) rely on precise  
25 gain (or scaling) in order to achieve satisfactory performance. Therefore, a scaling error  
compensator 1656 adjusts the gain of the receiver to compensate for variations in scaling.  
Further, a slicer 1658 then quantizes the scaled baseband symbols to the nearest ideal  
constellation points, which are the estimates of the symbols from the remote data pump  
transmitter (not shown). A decoder 1659 converts the output of slicer 1658 into a digital binary  
stream.

30 During data pump training, known transmitted training sequences are transmitted by a  
data pump transmitter in accordance with the applicable ITU-T standard. An ideal reference  
generator 1660, generates a local replica of the constellation point 1660(a). During the training  
phase a switch 1661 is toggled to connect the output 1660(a) of the ideal reference generator  
1660 to a difference operator 1662 that generates a baseband error signal 1662(a) by subtracting  
35 the ideal constellation sequence 1660(a) and the baseband equalizer output signal 1650(a). A  
carrier phase generator 1664 uses the baseband error signal 1662(a) and the baseband equalizer  
output signal 1650(a) to synchronize local carrier reference 1666 with the carrier of the received  
signal 1630(a). During the data phase the switch 1661 connects the output 1658(a) of the slicer

1 to the input of difference operator 1662 that generates a baseband error signal 1662(a) in the data  
 phase by subtracting the estimated symbol output by the slicer 1658 and the baseband equalizer  
 output signal 1650(a). It will be appreciated by one of skill that the described receiver is one of  
 several approaches. Alternate approaches in accordance with ITU-T recommendations may be  
 5 readily substituted for the described data pump. Accordingly, the described exemplary  
 embodiment of the data pump is by way of example only and not by way of limitation.

a. Timing Recovery System

10 Timing recovery refers to the process in a synchronous communication system whereby  
 timing information is extracted from the data being received. In the context of a modem  
 connection in accordance with an exemplary embodiment of the present invention, each modem  
 is coupled to a signal processing system, which for the purposes of explanation is operating in  
 a network gateway, either directly or through a PSTN line. In operation, each modem establishes  
 a modem connection with its respective network gateway, at which point, the modems begin  
 15 relaying data signals across a packet based network. The problem that arises is that the clock  
 frequencies of the modems are not identical to the clock frequencies of the data pumps operating  
 in their respective network gateways. By design, the data pump receiver in the network gateway  
 should sample a received signal of symbols in synchronism with the transmitter clock of the  
 modem connected locally to that gateway in order to properly demodulate the transmitted signal.

20 A timing recovery system can be used for this purpose. Although the timing recovery  
 system is described in the context of a data pump within a signal processing system with the  
 packet data modem exchange invoked, those skilled in the art will appreciate that the timing  
 recovery system is likewise suitable for various other applications in various other telephony and  
 telecommunications applications, including fax data pumps. Accordingly, the described  
 25 exemplary embodiment of the timing recovery system in a signal processing system is by way  
 of example only and not by way of limitation.

30 A block diagram of a timing recovery system is shown in FIG. 54. In the described  
 exemplary embodiment, the digital resampler 1640 resamples the gain adjusted signal 1636(a)  
 output by the AGC (see FIG. 53). A timing error estimator 1670 provides an indication of  
 whether the local timing or clock of the data pump receiver is leading or lagging the timing or  
 clock of the data pump transmitter in the local modem. As is known in the art, the timing error  
 estimator 1670 may be implemented by a variety of techniques including that proposed by  
 Godard. The A/D converter 1631 of the data pump receiver (see FIG. 53) samples the received  
 signal 1630(a) at a rate of  $f_0$  which is an integer multiple of the symbol rate  $f_s = 1/T$  and is at least  
 35 twice the highest frequency component of the received signal 1630(a) to satisfy nyquist sampling  
 theory. The samples are applied to an upper bandpass filter 1672 and a lower bandpass filter  
 1674. The upper bandpass filter 1672 is tuned to the upper bandedge frequency  $f_u = f_c + 0.5f_s$   
 and the lower bandpass filter 1674 is tuned to the lower bandedge frequency  $f_l = f_c - 0.5f_s$  where  
 $f_c$  is the carrier frequency of the QAM signal. The bandwidth of the filters 672 and 674 should

1 be reasonably narrow, preferably on the order of 100 Hz for a  $f_s = 2400$  baud modem. Conjugate  
logic 1676 takes the complex conjugate of complex output of the lower bandpass filter.  
Multiplier 1678 multiplies the complex output of the upper bandpass filter 1672(a) by the  
complex conjugate of the lower bandpass filter to form a cross-correlation between the output  
5 of the two filters (1672 and 1674). The real part of the correlated symbol is discarded by  
processing logic 1680, and a sampler 1681 samples the imaginary part of the resulting cross-  
correlation at the symbol rate to provide an indication of whether the timing phase error is leading  
or lagging.

10 In operation, a transmitted signal from a remote data pump transmitter (not shown)  $g(t)$   
is made to correspond to each data character. The signal element has a bandwidth approximately  
equal to the signaling rate  $f_s$ . The modulation used to transmit this signal element consists of  
multiplying the signal by a sinusoidal carrier of frequency  $f_c$  which causes the spectrum to be  
translated to a band around frequency  $f_c$ . Thus, the corresponding spectrum is bounded by  
frequencies  $f_1 = f_c - 0.5f_s$  and  $f_2 = f_c + 0.5f_s$ , which are known as the bandedge frequencies.  
15 Reference for more detailed information may be made to "Principles of Data Communication"  
by R. W. Lucky, J. Salz and E. J. Weldon, Jr., McGraw-Hill Book Company, pages 50-51.

In practice it has been found that additional filtering is required to reduce symbol clock  
jitter, particularly when the signal constellation contains many points. Conventionally a loop  
filter 1682 filters the timing recovery signal to reduce the symbol clock jitter. Traditionally the  
20 loop filter 1682 is a second order infinite impulse response (IIR) type filter, whereby the second  
order portion tracks the offset in clock frequency and the first order portion tracks the offset in  
phase. The output of the loop filter drives clock phase adjuster 1684. The clock phase adjuster  
controls the digital sampling rate of digital resampler 1640 so as to sample the received symbols  
in synchronism with the transmitter clock of the modem connected locally to that gateway.  
Typically, the clock phase adjuster 1684 utilizes a poly-phase interpolation algorithm to digitally  
25 adjust the timing phase. The timing recovery system may be implemented in either analog or  
digital form. Although digital implementations are more prevalent in current modem design an  
analog embodiment may be realized by replacing the clock phase adjuster with a VCO.

The loop filter 1682 is typically implemented as shown in FIG. 55. The first order portion  
30 of the filter controls the adjustments made to the phase of the clock (not shown). A multiplier  
1688 applies a first order adjustment constant  $\alpha$  to advance or retard the clock phase adjustment.  
Typically the constant  $\alpha$  is empirically derived via computer simulation or a series of simple  
experiments with a telephone network simulator. Generally  $\alpha$  is dependent upon the gain and  
the bandwidth of the upper and lower filters in the timing error estimator, and is generally  
optimized to reduce symbol clock jitter and control the speed at which the phase is adjusted. The  
35 structure of the loop filter 1682 may include a second order component 1690 that estimates the  
offset in clock frequency. The second order portion utilizes an accumulator 1692 in a feedback  
loop to accumulate the timing error estimates. A multiplier 1694 is used to scale the accumulated  
timing error estimate by a constant  $\beta$ . Typically, the constant  $\beta$  is empirically derived based on

1 the amount of feedback that will cause the system to remain stable. Summer 1695 sums the  
scaled accumulated frequency adjustment 1694(a) with the scaled phase adjustment 1688(a). A  
disadvantage of conventional designs which include a second order component 1690 in the loop  
filter 1682 is that such second order components 1690 are prone to instability with large  
5 constellation modulations under certain channel conditions.

An alternative digital implementation eliminates the loop filter. Referring to FIG. 56 a  
hard limiter 1695 and a random walk filter 1696 are coupled to the output of the timing error  
estimator 1680 to reduce timing jitter. The hard limiter 1695 provides a simple automatic gain  
control action that keeps the loop gain constant independent of the amplitude level of the input  
10 signal. The hard limiter 1695 assures that timing adjustments are proportional to the timing of  
the data pump transmitter of the local modem and not the amplitude of the received signal. The  
random walk filter 1696 reduces the timing jitter induced into the system as disclosed in  
"Communication System Design Using DSP Algorithms", S. Tretter, p. 132, Plenum Press, NY.,  
1995, the contents of which is hereby incorporated by reference as through set forth in full herein.  
15 The random walk filter 1696 acts as an accumulator, summing a random number of adjustments  
over time. The random walk filter 1696 is reset when the accumulated value exceeds a positive  
or negative threshold. Typically, the sampling phase is not adjusted so long as the accumulator  
output remains between the thresholds, thereby substantially reducing or eliminating incremental  
positive adjustments followed by negative adjustments that otherwise tend to not accumulate.

20 Referring to FIG. 57 in an exemplary embodiment of the present invention, the multiplier  
1688 applies the first order adjustment constant  $\alpha$  to the output of the random walk filter to  
advance or retard the estimated clock phase adjustment. In addition, a timing frequency offset  
compensator 1697 is coupled to the timing recovery system via switches 1698 and 1699 to  
preferably provide a fixed dc component to compensate for clock frequency offset present in the  
25 received signal. The exemplary timing frequency offset compensator preferably operates in  
phases. A frequency offset estimator 1700 computes the total frequency offset to apply during  
an estimation phase and incremental logic 1701, incrementally applies the offset estimate in  
linear steps during the application phase. Switch control logic 1702 controls the toggling of  
switches 1698 and 1699 during the estimation and application phases of compensation  
adjustment. Unlike the second order component 1690 of the conventional timing recovery loop  
30 filter disclosed in FIG. 55, the described exemplary timing frequency offset compensator 1697  
is an open loop design such that the second order compensation is fixed during steady state.  
Therefore, switches 1698 and 1699 work in opposite cooperation when the timing compensation  
is being estimated and when it is being applied.

35 During the estimation phase, switch control logic 1702 closes switch 1698 thereby  
coupling the timing frequency offset compensator 1697 to the output of the random walk filter  
1696, and opens switch 1699 so that timing adjustments are not applied during the estimation  
phase. The frequency offset estimator 1700 computes the timing frequency offset during the  
estimation phase over K symbols in accordance with the block diagram shown in FIG. 58. An



1 accumulator 1703 accumulates the frequency offset estimates over K symbols. A multiplier 1704  
is used to average the accumulated offset estimate by applying a constant  $\gamma/K$ . Typically the  
constant  $\gamma$  is empirically derived and is preferably in the range of about 0.5-2. Preferably K is  
as large as possible to improve the accuracy of the average. K is typically greater than about 500  
5 symbols and less than the recommended training sequence length for the modem in question. In  
the exemplary embodiment the first order adjustment constant  $\alpha$  is preferably in the range of  
about 100-300 part per million (ppm). The timing frequency offset is preferably estimated during  
the timing training phase (timing tone) and equalizer training phase based on the accumulated  
adjustments made to the clock phase adjuster 1684 over a period of time.

10 During steady state operation when the timing adjustments are applied, switch control  
logic 1702 opens switch 1698 decoupling the timing frequency offset compensator 1697 from  
the output of the random walk filter, and closes switch 1699 so that timing adjustments are  
applied by summer 1705. After K symbols of a symbol period have elapsed and the frequency  
offset compensation is computed, the incremental logic 1701 preferably applies the timing  
15 frequency offset estimate in incremental linear steps over a period of time to avoid large sudden  
adjustments which may throw the feedback loop out of lock. This is the transient phase. The  
length of time over which the frequency offset compensation is incrementally applied is  
empirically derived, and is preferably in the range of about 200-800 symbols. After the  
incremental logic 1701 has incrementally applied the total timing frequency offset estimate  
computed during the estimate phase, a steady state phase begins where the compensation is fixed.  
20 Relative to conventional second order loop filters, the described exemplary embodiment provides  
improved stability and robustness.

#### b. Multipass Training

25 Data pump training refers to the process by which training sequences are utilized to train  
various adaptive elements within a data pump receiver. During data pump training, known  
transmitted training sequences are transmitted by a data pump transmitter in accordance with the  
applicable ITU-T standard. In the context of a modem connection in accordance with an  
exemplary embodiment of the present invention, the modems (see FIG. 49) are coupled to a  
signal processing system, which for the purposes of explanation is operating in a network  
30 gateway, either directly or through a PSTN line. In operation, the receive data pump operating  
in each network gateway of the described exemplary embodiment utilizes PSFSE architecture.  
The PSFSE architecture has numerous advantages over other architectures when receiving QAM  
signals. However, the PSFSE architecture has a slow convergence rate when employing the least  
mean square (LMS) stochastic gradient algorithm. This slow convergence rate typically prevents  
the use of PSFSE architecture in modems that employ relatively short training sequences in  
35 accordance with common standards such as V.29. Because of the slow convergence rate, the  
described exemplary embodiment re-processes blocks of training samples multiple times (multi-  
pass training).

1 Although the method of performing multi-pass training is described in the context of a  
signal processing system with the packet data exchange invoked, those skilled in the art will  
appreciate that multi-pass training is likewise suitable for various other telephony and  
telecommunications applications. Accordingly, the described exemplary method for multi-pass  
5 training in a signal processing system is by way of example only and not by way of limitation.

In an exemplary embodiment the data pump receiver operating in the network gateway  
stores the received QAM samples of the modem's training sequence in a buffer until N symbols  
have been received. The PSFSE is then adapted sequentially over these N symbols using a LMS  
algorithm to provide a coarse convergence of the PSFSE. The coarsely converged PSFSE (i.e.  
10 with updated values for the equalizer taps) returns to the start of the same block of training  
samples and adapts a second time. This process is repeated M times over each block of training  
samples. Each of the M iterations provides a more precise or finer convergence until the PSFSE  
is completely converged.

15 c. Scaling Error Compensator

Scaling error compensation refers to the process by which the gain of a data pump  
receiver (fax or modem) is adjusted to compensate for variations in transmission channel  
conditions. In the context of a modem connection in accordance with an exemplary embodiment  
of the present invention, each modem is coupled to a signal processing system, which for the  
20 purposes of explanation is operating in a network gateway, either directly or through a PSTN line.  
In operation, each modem communicates with its respective network gateway using digital  
modulation techniques. The problem that arises is that digital modulation techniques such as  
QAM and pulse amplitude modulation (PAM) rely on precise gain (or scaling) in order to achieve  
satisfactory performance. In addition, transmission in accordance with the V.34  
25 recommendations typically includes a training phase and a data phase whereby a much smaller  
constellation size is used during the training phase relative to that used in the data phase. The  
V.34 recommendation, requires scaling to be applied when switching from the smaller  
constellation during the training phase into the larger constellation during the data phase.

The scaling factor can be precisely computed by theoretical analysis, however, different  
30 manufacturers of V.34 systems (modems) tend to use slightly different scaling factors. Scaling  
factor variation (or error) from the predicted value may degrade performance until the PSFSE  
compensates for the variation in scaling factor. Variation in gain due to transmission channel  
conditions is compensated by an initial gain estimation algorithm (typically consisting of a simple  
signal power measurement during a particular signaling phase) and an adaptive equalizer during  
the training phase. However, since a PSFSE is preferably configured to adapt very slowly during  
35 the data phase, there may be a significant number of data bits received in error before the PSFSE  
has sufficient time to adapt to the scaling error.

1 It is, therefore, desirable to quickly reduce the scaling error and hence minimize the  
number of potential erred bits. A scaling factor compensator can be used for this purpose.  
Although the scaling factor compensator is described in the context of a signal processing system  
with the packet data modem exchange invoked, those skilled in the art will appreciate that the  
5 preferred scaling factor compensator is likewise suitable for various other telephony and  
telecommunications applications. Accordingly, the described exemplary embodiment of the  
scaling factor compensator in a signal processing system is by way of example only and not by  
way of limitation.

10 FIG. 59 shows a block diagram of an exemplary embodiment of the scaling error  
compensator in a data pump receiver 1630 (see FIG. 53). In an exemplary embodiment, scaling  
error compensator 1708 computes the gain adjustment of the data pump receiver. Multiplier  
1710 adjusts a nominal scaling factor 1712 (the scaling error computed by the data pump  
manufacturer) by the gain adjustment as computed by the scaling error compensator 1708. The  
combined scale factor 1710(a) is applied to the incoming symbols by multiplier 1714. A slicer  
1716 quantizes the scaled baseband symbols to the nearest ideal constellation points, which are  
15 the estimates of the symbols from the remote data pump transmitter.

The scaling error compensator 1708 preferably includes a divider 1718 which estimates  
the gain adjustment of the data pump receiver by dividing the expected magnitude of the received  
symbol 1716(a) by the actual magnitude of the received symbol 1716(b). In the described  
20 exemplary embodiment the magnitude is defined as the sum of squares between real and  
imaginary parts of the complex symbol. The expected magnitude of the received symbol is the  
output 1716(a) of the slicer 1716 (i.e. the symbol quantized to the nearest ideal constellation  
point) whereas the magnitude of the actual received symbol is the input 1716(b) to the slicer  
1716. In the case where a Viterbi decoder performs the error-correction of the received,  
noise-disturbed signal (as for V.34), the output of the slicer may be replaced by the first level  
25 decision of the Viterbi decoder.

The statistical nature of noise is such that large spikes in the amplitude of the received  
signal will occasionally occur. A large spike in the amplitude of the received signal may result  
in an erroneously large estimate of the gain adjustment of the data pump receiver. Typically,  
30 scaling is applied in a one to one ratio with the estimate of the gain adjustment, so that large  
scaling factors may be erroneously applied when large amplitude noise spikes are received. To  
minimize the impact of large amplitude spikes and improve the accuracy of the system, the  
described exemplary scaling error compensator 1708 further includes a non-linear filter in the  
form of a hard-limiter 1720 which is applied to each estimate 1718(a). The hard limiter 1720  
limits the maximum adjustment of the scaling value. The hard limiter 1720 provides a simple  
35 automatic control action that keeps the loop gain constant independent of the amplitude of the  
input signal so as to minimize the negative effects of large amplitude noise spikes. In addition,  
averaging logic 1722 computes the average gain adjustment estimate over a number (N) of  
symbols in the data phase prior to adjusting the nominal scale factor 1710. As will be

1 appreciated by those of skill in the art, other non-linear filtering algorithms may also be used in place of the hard-limiter.

5 Alternatively, the accuracy of the scaling error compensation may be further improved by estimating the averaged scaling adjustment twice and applying that estimate in two steps. A large hard limit value (typically  $1 \pm 0.25$ ) is used to compute the first average scaling adjustment. The initial prediction provides an estimate of the average value of the amplitude of the received symbols. The unpredictable nature of the amplitude of the received signal requires the use of a large initial hard limit value to ensure that the true scaling error is included in the initial estimate of the average scaling adjustment. The estimate of the average value of the amplitude of the received symbols is used to calibrate the limits of the scaling adjustment. The average scaling adjustment is then estimated a second time using a lower hard limit value and then applied to the nominal scale factor 1712 by multiplier 1710.

15 In most modem specifications, such as the V.34 standards, there is a defined signaling period (B1 for V.34) after transition into data phase where the data phase constellation is transmitted with signaling information to flush the receiver pipeline (i.e. Viterbi decoder etc.) prior to the transmission of actual data. In an exemplary embodiment this signaling period may be used to make the scaling adjustment such that any scaling error is compensated for prior to actual transfer of data.

20 d. Non-Linear Decoder

25 In the context of a modem connection in accordance with an exemplary embodiment of the present invention, each modem is coupled to a signal processing system, which for the purposes of explanation is operating in a network gateway, either directly or through a PSTN line. In operation, each modem communicates with its respective network gateway using digital modulation techniques. The international telecommunications union (ITU) has promulgated standards for the encoding and decoding of digital data in ITU-T Recommendation G.711 (ref. G.711) which is incorporated herein by reference as if set forth in full. The encoding standard specifies that a nonlinear operation (companding) be performed on the analog data signal prior to quantization into seven bits plus a sign bit. The companding operation is a monatomic invertible function which reduces the higher signal levels. At the decoder, the inverse operation (expanding) is done prior to analog reconstruction. The companding / expanding operation quantizes the higher signal values more coarsely. The companding / expanding operation, is suitable for the transmission of voice signals but introduces quantization noise on data modem signals. The quantization error (noise) is greater for the outer signal levels than the inner signal levels.

35 The ITU-T Recommendation V.34 describes a mechanism whereby (ref. V.34) the uniform signal is first expanded (ref. BETTS) to space the outer points farther apart than the inner points before G.711 encoding and transmission over the PCM link. At the receiver, the inverse

1 operation is applied after G.711 decoding. The V.34 recommended expansion/inverse operation  
yields a more uniform signal to noise ratio over the signal amplitude. However, the inverse  
operation specified in the ITU-T Recommendation V.34 requires a complex receiver calculation.  
The calculation is computationally intensive, typically requiring numerous machine cycles to  
5 implement.

It is, therefore, desirable to reduce the number of machine cycles required to compute the  
inverse to within an acceptable error level. A simplified nonlinear decoder can be used for this  
purpose. Although the nonlinear decoder is described in the context of a signal processing  
system with the packet data modem exchange invoked, those skilled in the art will appreciate that  
10 the nonlinear decoder is likewise suitable for various other telephony and telecommunications  
application. Accordingly, the described exemplary embodiment of the nonlinear decoder in a  
signal processing system is by way of example only and not by way of limitation.

Conventionally, iteration algorithms have been used to compute the inverse of the G.711  
nonlinear warping function. Typically, iteration algorithms generate an initial estimate of the  
15 input to the nonlinear function and then compute the output. The iteration algorithm compares  
the output to a reference value and adjusts the input to the nonlinear function. A commonly used  
adjustment is the successive approximation wherein the difference between the output and the  
reference function is added to the input. However, when using the successive approximation  
technique, up to ten iterations may be required to adjust the estimated input of the nonlinear  
20 warping function to an acceptable error level, so that the nonlinear warping function must be  
evaluated ten times. The successive approximation technique is computationally intensive,  
requiring significant machine cycles to converge to an acceptable approximation of the inverse  
of the nonlinear warping function. Alternatively, a more complex warping function is a linear  
Newton Rhapson iteration. Typically the Newton Rhapson algorithm requires three evaluations  
25 to converge to an acceptable error level. However, the inner computations for the Newton  
Rhapson algorithm are more complex than those required for the successive approximation  
technique. The Newton Rhapson algorithm utilizes a computationally intensive iteration loop  
wherein the derivative of the nonlinear warping function is computed for each approximation  
iteration, so that significant machine cycles are required to conventionally execute the Newton  
Rhapson algorithm.

30 An exemplary embodiment of the present invention modifies the successive  
approximation iteration. A presently preferred algorithm computes an approximation to the  
derivative of the nonlinear warping function once before the iteration loop is executed and uses  
the approximation as a scale factor during the successive approximation iterations. The described  
exemplary embodiment converges to the same acceptable error level as the more complex  
35 conventional Newton-Rhapson algorithm in four iterations. The described exemplary  
embodiment further improves the computational efficiency by utilizing a simplified  
approximation of the derivative of the nonlinear warping function.

In operation, development of the described exemplary embodiment proceeds as follows with a warping function defined as:

$$w(v) = \frac{\Theta(v)}{6} + \frac{\Theta(v)^2}{120}$$

the V.34 nonlinear decoder can be written as

$$Y = X(1 + w(\|X\|^2))$$

taking the square of the magnitude of both sides yields,

$$Y^2 = |X|^2 (1 + w(\|X\|^2))^2$$

The encoder notation can then be simplified with the following substitutions

$$Y_r = \|Y\|^2, X_r = \|X\|^2$$

and write the V.34 nonlinear encoder equation in the canonical form  $G(x)=0$ .

$$X_r(1 + w(X_r))^2 - Y_r = 0$$

The Newton-Rhapson iteration is a numerical method to determine  $X$  that results in an iteration of the form:

$$X_{n+1} = X_n - \frac{G(X_n)}{G'(X_n)}$$

where  $G'$  is the derivative and the substitution iteration results when  $G'$  is set equal to one.

The computational complexity of the Newton-Rhapson algorithm is thus paced by the derivation of the derivative  $G'$ , which conventionally is related to  $X_r$  so that the mathematical instructions saved by performing fewer iterations are offset by the instructions required to calculate the derivative and perform the divide. Therefore, it would be desirable to approximate the derivative  $G'$  with a term that is the function of the input  $Y_r$  so that  $G(x)$  is a monotonic function and  $G'(x)$  can be expressed in terms of  $G(x)$ . Advantageously, if the steps in the iteration are small, then  $G'(x)$  will not vary greatly and can be held constant over the iteration. A series of simple experiments yields the following approximation of  $G'(x)$  where  $\alpha$  is an experimentally derived scaling factor.

$$G' = \frac{1+\alpha Y_r}{\alpha}$$

1           The approximation for  $G'$  converges to an acceptable error level in a minimum number of steps, typically one more iteration than the full linear Newton-Rhapson algorithm. A single divide before the iteration loop computes the quantity

$$\frac{1}{G'} = \frac{\alpha}{1+Yr}$$

5           The error term is multiplied by  $1/G'$  in the successive iteration loop. It will be appreciated by one of skill in the art that further improvements in the speed of convergence are possible with the "Generalized Newton-Rhapson" class of algorithms. However, the inner loop computations for this class of algorithm are quite complex.

10           Advantageously, the described exemplary embodiment does not expand the polynomial because the numeric quantization on a store in a sixteen bit machine may be quite significant for the higher order polynomial terms. The described exemplary embodiment organizes the inner loop computations to minimize the effects of truncation and the number of instructions required for execution. Typically the inner loop requires eighteen instructions and four iterations to converge to within two bits of the actual value which is within the computational roundoff noise of a sixteen bit machine.

#### D.    Human Voice Detector

20           In a preferred embodiment of the present invention, a signal processing system is employed to interface telephony devices with packet based networks. Telephony devices include, by way of example, analog and digital phones, ethernet phones, Internet Protocol phones, fax machines, data modems, cable voice modems, interactive voice response systems, PBXs, key systems, and any other conventional telephony devices known in the art. In the described exemplary embodiment the packet voice exchange is common to both the voice mode and the voiceband data mode. In the voiceband data mode, the network VHD invokes the packet voice exchange for transparently exchanging data without modification (other than packetization) between the telephony device or circuit switched network and the packet based network. This is typically used for the exchange of fax and modem data when bandwidth concerns are minimal as an alternative to demodulation and remodulation.

30           During the voiceband data mode, the human voice detector service is also invoked by the resource manager. The human voice detector monitors the signal from the near end telephony device for voice. The described exemplary human voice detector estimates pitch period of an incoming telephony signal and compares the pitch period of said telephony signal to a plurality of thresholds to identify active voice samples. This approach is substantially independent of the amplitude of the spoken utterance, so that whispered or shouted utterance may be accurately identified as active voice samples. In the event that voice is detected by the human voice detector, an event is forwarded to the resource manager which, in turn, causes the resource manager to

1 terminate the human voice detector service and invoke the appropriate services for the voice mode (i.e., the call discriminator, the packet tone exchange, and the packet voice exchange).

5 Although a preferred embodiment is described in the context of a signal processing system for telephone communications across the packet based network, it will be appreciated by those skilled in the art that the voice detector is likewise suitable for various other telephony and telecommunications application. Accordingly, the described exemplary embodiment of the voice detector in a signal processing system is by way of example only and not by way of limitation.

10 There are a variety of encoding methods known for encoding voice. Most frequently, voice is modeled on a short-time basis as the response of a linear system excited by a periodic impulse train for voiced sounds or random noise for the unvoiced sounds. Conventional human voice detectors typically monitor the power level of the incoming signal to make a voice / machine decision. Typically, if the power level of the incoming signal is above a predetermined threshold, the sequence is typically declared voice. The performance of such conventional voice  
15 detectors may be degraded by the environment, in that a very soft spoken whispered utterance will have a very different power level from a loud shout. If the threshold is set at too low a level, noise will be declared voice, whereas if the threshold is set at too high a level a soft spoken voice segment will be incorrectly marked as inactive.

20 Alternatively, voice may generally be classified as voiced if a fundamental frequency is imported to the air stream by the vocal cords of the speaker. In such case, the frequency of a voice segment is typically highly periodic at around the pitch frequency. The determination as to whether a voice segment is voiced or unvoiced, and the estimation of the fundamental frequency can be obtained in a variety of ways known in the art such as pitch detection algorithms. In the described exemplary embodiment, the human voice detector calculates an  
25 autocorrelation function for the incoming signal. An autocorrelation function for a voice segment demonstrates local peaks with a periodicity in proportion to the pitch period. The human voice detector service utilizes this feature in conjunction with power measurements to distinguish voice signals from modem signals. It will be appreciated that other pitch detection algorithms known in the art can be used as well.

30 Referring to FIG. 60, in the described exemplary embodiment, a power estimator 1730 estimates the power level of the incoming signal. Autocorrelation logic 1732 computes an autocorrelation function for an input signal to assist in the voice/machine decision. Autocorrelation, as is known in the art, involves correlating a signal with itself. A correlation  
35 function shows how similar two signals are, and how long the signals remain similar when one is shifted with respect to the other. Periodic signals go in and out of phase as one is shifted with respect to the other, so that a periodic signal will show strong correlation at shifts where the peaks coincide. Thus, the autocorrelation of a periodic signal is itself a periodic signal, with a period equal to the period of the original signal.



1           The autocorrelation calculation computes the autocorrelation function over an interval of 360 samples with the following approach:

$$R[k] = \sum_{n=0}^{N-k-1} x[n]x[n+k]$$

5       where  $N=360$ ,  $k=0,1,2,...179$ .

10           A pitch tracker 1734 estimates the period of the computed autocorrelation function. Framed based decision logic 1736 analyzes the estimated power level 1730a, the autocorrelation function 1732a and the periodicity 1734a of the incoming signal to execute a frame based voice/machine decision according to a variety of factors. For example, the energy of the input signal should be above a predetermined threshold level, preferably in the range of about -45 to -55 dBm, before the frame based decision logic 1736 declares the signal to be voice. In addition, the typical pitch period of a voice segment should be in the range of about 60-400 Hz, so that the autocorrelation function should preferably be periodic with a period in the range of about 60-400 Hz before the frame based decision logic 1736 declares a signal as active or containing voice.

15           The amplitude of the autocorrelation function is a maximum for  $R[0]$ , i.e. when the signal is not shifted relative to itself. Also, for a periodic voice signal, the amplitude of the autocorrelation function with a one period shift (i.e.  $R[\text{pitch period}]$ ) should preferably be in the range of about 0.25-0.40 of the amplitude of the autocorrelation function with no shift (i.e.  $R[0]$ ).  
20       Similarly, modem signaling may involve certain DTMF or MF tones, in this case the signals are highly correlated, so that if the largest peak in the amplitude of the autocorrelation function after  $R[0]$  is relatively close in magnitude to  $R[0]$ , preferably in the range of about 0.75-0.90  $R[0]$ , the frame based decision logic 1736 declares the sequence as inactive or not containing voice.

25           Once a decision is made on the current frame as to voice or machine, final decision logic 1738 compares the current frame decision with the two adjacent frame decisions. This check is known as backtracking. If a decision conflicts with both adjacent decisions it is flipped, i.e. voice decision turned to machine and vice versa.

30           Although a preferred embodiment of the present invention has been described, it should not be construed to limit the scope of the appended claims. For example, the present invention can be implemented by both a software embodiment or a hardware embodiment. Those skilled in the art will understand that various modifications may be made to the described embodiment. Moreover, to those skilled in the various arts, the invention itself herein will suggest solutions to other tasks and adaptations for other applications. It is therefore desired that the present  
35       embodiments be considered in all respects as illustrative and not restrictive, reference being made to the appended claims rather than the foregoing description to indicate the scope of the invention.

## 1 WHAT IS CLAIMED IS:

1. A cable modem, comprising:  
a network port to interface with a packet based network;  
5 a telephony port to interface with a telephony device;  
a data port to interface with a data terminal; and  
a processor coupled to each of the ports.
2. The cable modem of claim 1 further comprising a transceiver coupled between  
the processor and the network port.
3. The cable modem of claim 2 wherein the transceiver comprises a media access  
10 controller (MAC) coupled to the processor, and a modulator and a demodulator both disposed  
between the MAC and the network port.
4. The cable modem of claim 4 wherein the modulator and the demodulation each  
15 comprises quadrature amplitude modulation.
5. The cable modem of claim 1 wherein the processor determines whether voices  
signals from the network port are destined for the data port or the telephony port and couples the  
voice signals to one of the data port and telephony port based on such determination.
6. The cable modem of claim 1 further comprising a voice circuit coupled between  
20 the telephony port and the processor.
7. The cable modem of claim 6 wherein the processor formats voice signals flowing  
from the telephony port to the processor into voice signal packets, and formats voice signals  
25 flowing from the processor to the telephony port into a telephony format.
8. The cable modem of claim 7 wherein the telephony format comprises pulse code  
modulation.
9. The cable modem of claim 6 wherein the voice circuit comprises a jitter buffer  
30 to receive voice signal packets of varying delay from the processor and compensating for the  
delay variation of the received voice signal packets.
10. The cable modem of claim 9 wherein the jitter buffer comprises a voice queue  
35 which buffers the received voice signals for a holding time, and a voice synchronizer which  
adaptively adjusts the holding time of the voice queue.
11. The cable modem of claim 6 wherein the voice circuit comprises a tone exchange  
to exchange DTMF signals between the telephony port and the processor.

1  
5  
12. The cable modem of claim 6 wherein the voice circuit comprises a voice decoder to decode packets of voice signal flowing from the processor to the telephony port, a voice activity detector to detect the voice signals without speech, and a comfort noise generator to insert comfort noise in place of the voice signals without speech.

10  
13. The cable modem of claim 6 wherein the voice circuit comprises a voice decoder to decode packets of the voice signals flowing from the processor to the telephony port, a voice activity detector to detect lost voice signals, and a lost packet recovery engine to process the voice signals to compensate for the lost voice signals.

14. The cable modem of claim 6 wherein the voice circuit comprises a voice encoder to encode the voice signals flowing from the telephony port to the processor, and a voice activity detector to suppresses the voice signals without speech.

15  
15. The cable modem of claim 14 wherein the voice circuit further comprises a comfort noise estimator to generate comfort noise parameters when the voice activity detector suppresses the voice signals without speech.

20  
16. The cable modem of claim 6 wherein the voice circuit further comprises a decoder to decode packets of the voice signals flowing from the processor to the telephony port, and an echo canceller capable of cancelling decoded voice signal echos on voice signals flowing from the telephony port to the processor.

25  
17. A gateway, comprising:  
a network port to interface with a packet based network;  
a telephony port to interface to a telephony device;  
a data port to interface to a data terminal;  
a processor coupled to each of the ports; and  
a transceiver disposed between the processor and the network port, the transceiver being capable of transmitting and receiving packets of voice signals, packets of data signals, and a television signal.

30  
18. The gateway of claim 17 wherein the transceiver comprises a media access controller (MAC) coupled to the processor, and a modulator and a demodulator both disposed between the MAC and the network port.

35  
19. The gateway of claim 18 wherein the modulator and the demodulation each comprises quadrature amplitude modulation.

1           20.    The gateway of claim 17 wherein the processor determines whether the voices  
signals from the network port are destined for the data port or the telephony port and couples the  
voice signals to one of the data port and telephony port based on such determination.

5           21    The gateway of claim 17 further comprising a voice circuit coupled between the  
telephony port and the processor.

10           22.    The gateway of claim 21 wherein the processor formats voice signals flowing  
from the telephony port to the processor into packets of voice signals, and formats the packets  
of voice signals flowing from the processor to the telephony port into a telephony format.

15           23.    The gateway of claim 22 wherein the telephony format comprises pulse code  
modulation.

20           24.    The gateway of claim 21 wherein the voice circuit comprises a jitter buffer to  
receive the packets of voice signals of varying delay from the processor and compensating for  
the delay variation of the received packets.

25           25.    The gateway of claim 24 wherein the jitter buffer comprises a voice queue which  
buffers the received voice signals for a holding time, and a voice synchronizer which adaptively  
adjusts the holding time of the voice queue.

30           26.    The gateway of claim 21 wherein the voice circuit comprises a tone exchange to  
exchange DTMF signals between the telephony port and the processor.

35           27.    The gateway of claim 21 wherein the voice circuit comprises a voice decoder to  
decode packets of voice signals flowing from the processor to the telephony port, a voice activity  
detector to detect the voice signals without speech, and a comfort noise generator to insert  
comfort noise in place of the voice signals without speech.

40           28.    The gateway of claim 21 wherein the voice circuit comprises a voice decoder to  
decode packets of the voice signals flowing from the processor to the telephony port, a voice  
activity detector to detect lost voice signals, and a lost packet recovery engine to process the  
voice signals to compensate for the lost voice signals.

45           29.    The gateway of claim 21 wherein the voice circuit comprises a voice encoder to  
encode the voice signals flowing from the telephony port to the processor, and a voice activity  
detector to suppresses the voice signals without speech.

50           30.    The gateway of claim 29 wherein the voice circuit further comprises a comfort  
noise estimator to generate comfort noise parameters when the voice activity detector suppresses  
the voice signals without speech.

1           31.     The gateway of claim 21 wherein the voice circuit further comprises a decoder to decode packets of the voice signals flowing from the processor to the telephony port, and an echo canceller capable of cancelling decoded voice signal echos on voice signals flowing from the telephony port to the processor.

5           32.     A method of processing signals, comprising:  
              exchanging voice signals between a packet based network and a telephony device;  
              exchanging data signals between the packet based network and a data terminal;  
and  
              receiving a television signal from the packet based network.

10          33.     The method of claim 32 further comprising demodulating the signals flowing from the packet based network to the telephony device and the data terminal, and modulating the signals flowing from the telephony device and the data terminal to the packet based network.

15          34     The method of claim 33 wherein the modulator and the demodulation of the signals each comprises quadrature amplitude modulation.

          35.     The method of claim 32 wherein the data signal exchange comprises receiving and transmitting the data signals to the data terminal over a local area network (LAN).

20          36.     The method of claim 35 wherein the voice signal exchange comprises receiving and transmitting the voice signals to the telephony device over a telephone line.

25          37.     The method of claim 36 wherein the voice signal exchange further comprises exchanging second voice signals between the packet based network and a second telephony, the second voice signal exchange comprising receiving and transmitting the second voice signals to the second telephony device over the LAN.

30          38.     The method of claim 32 wherein the voice exchange comprises formatting the voice signals flowing from the telephony device to the packet based network into voice signal packets, and formatting the voice signals flowing from the packet based network to the telephony device into a telephony format.

          39.     The method of claim 38 wherein the telephony format comprises pulse code modulation.

35          40.     The method of claim 32 wherein the voice signal exchange comprises receiving packets of voice signals of varying delay, buffering the received packets, and compensating for the delay variation of the received packets.

1           41.     The method of claim 32 further comprising exchanging DTMF signals between  
the packet based network and the telephony device.

5           42.     The method of claim 32 wherein the voice signal exchange comprises detecting  
the voice signals flowing from the packet based network to the telephony device not comprising  
speech, and inserting comfort noise in place of the non-speech voice signals.

10          43.     The method of claim 32 wherein the voice signal exchange comprises processing  
the voice signals flowing from the packet based network to the telephony device to compensate  
for the lost voice signals.

15          44.     The method of claim 32 wherein the voice signal exchange comprises suppressing  
voice signals without speech flowing from the telephony device to the packet based network.

20          45.     The method of claim 44 wherein the voice signal exchange further comprises  
circuit further comprises transmitting comfort noise parameters to the packet based network when  
the voice signals without speech are suppressed.

25          46.     The method of claim 32 wherein the voice signal exchange comprises decoding  
the voice signals flowing from the packet based network to the telephony device, and cancelling  
decoded voice signal echos on the voice signals flowing from the telephony device to the packet  
based network.

30          47.     A method of conditioning a composite signal, the composite signal being formed  
by introducing a first signal into a second signal, comprising:  
            adaptively filtering the first signal;  
            detecting the second signal in the composite signal;  
            controlling filter adaptation of the first signal responsive to detecting the second signal  
in the composite signal; and  
            conditioning the composite signal after the filter adaptation is controlled.

35          48.     The method of claim 47 wherein the composite signal conditioning comprises  
subtracting the adaptively filtered first signal from the composite signal.

40          49.     The method of claim 47 wherein the composite signal conditioning comprises  
delaying the composite signal conditioning from the detection of the second signal in the  
composite signal.

45          50.     The method of claim 47 wherein the detection of the second signal comprises  
estimating an absolute value of the first signal, estimating a maximum of the composite signal,  
comparing the absolute value of the first signal to the estimated maximum of the composite  
signal, and detecting said second signal as a function of the comparison.

1           51.    The method of claim 50 wherein the maximum estimation comprises applying a  
weighting function to a plurality of composite signal samples, estimating the absolute value of  
the weighted samples, determining the maximum of the weighted samples, comparing the  
maximum of the weighted samples to a previous maximum estimation, and selecting the  
5           maximum of the weighted samples as the maximum estimation when the maximum of the  
weighted samples is greater than the previous maximum estimation.

10           52.    The method of claim 51 further comprising applying a scale factor to the previous  
maximum estimation when the estimated maximum of the weighted samples is less than the  
previous maximum estimation, and selecting the scaled previous maximum estimation as the  
maximum estimation.

15           53.    The method of claim 47 further comprising pre-conditioning the composite signal  
prior to the detection of the second signal thereon.

20           54.    The method of claim 53 wherein pre-conditioning the composite signal comprises  
adaptively filtering the first signal and subtracting the adaptively filtered first signal from the  
composite signal.

25           55.    The method of claim 47 wherein the adaptive filtering is based on adaptation  
coefficients and filter adaptation control comprises holding the adaptation coefficients constant  
when the second signal is detected in the composite signal.

30           56.    A method of cancelling a far end echo from a near end signal, comprising:  
adaptively filtering a far end signal based on adaptation coefficients;  
detecting whether voice is present on the near end signal;  
holding the adaptation coefficients constant when voice is present; and  
25           cancelling the echo from the near end signal after the adaptation coefficients are held  
constant.

35           57.    The method of claim 56 wherein the echo cancellation comprises subtracting the  
adaptively filtered far end signal from the near end signal.

40           58.    The method of claim 56 wherein the echo cancellation comprises delaying the  
cancellation of the echo from the near end signal from the detection of voice thereon.

45           59.    The method of claim 56 wherein the detection of voice comprises estimating an  
absolute value of the far end signal, estimating a maximum of the near end signal, comparing the  
absolute value of the far end signal to the estimated maximum of the near end signal, and  
detecting voice on the near end signal as a function of the comparison.

1           60.     The method of claim 59 wherein the maximum estimation comprises applying a  
weighting function to a plurality of near end signal samples, estimating the absolute value of the  
weighted samples, determining the maximum of the weighted samples, comparing the maximum  
5     of the weighted samples to a previous maximum estimation, and selecting the maximum of the  
weighted samples as the maximum estimation when the maximum of the weighted samples is  
greater than the previous maximum estimation.

10           61.     The method of claim 60 further comprising applying a scale factor to the previous  
maximum estimation when the estimated maximum of the weighted samples is less than the  
previous maximum estimation, and selecting the scaled previous maximum estimation as the  
maximum estimation.

          62.     The method of claim 56 further comprising pre-conditioning the near end signal  
prior to the detection of voice thereon.

15           63.     The method of claim 62 wherein pre-conditioning the near end signal comprises  
adaptively filtering the far end signal and subtracting the adaptively filtered far end signal from  
the near end signal.

20           64.     A signal conditioner for conditioning composite signal, the composite signal being  
formed by introducing a first signal into a second signal, comprising:  
logic to detect the second signal in the composite signal;  
an adaptive filter to filter the first signal, wherein the logic controls filter adaptation  
responsive to detecting the second signal in the composite signal;  
a difference operator to subtract the filtered first signal from the composite signal; and  
a delay between the logic and the difference operator.

25           65.     The signal conditioner of claim 64 wherein the delay comprises a filter.

          66.     The signal conditioner of claim 64 wherein the delay comprises a decimator.

30           67.     The signal conditioner of claim 64 wherein the delay comprises a buffer.

          68.     The signal conditioner of claim 64 further comprising a first estimator to estimate  
an absolute value of the first signal, and a second estimator to estimate a maximum of the  
composite signal, wherein the logic compares the absolute value of the first signal to the  
estimated maximum of the composite signal and detects the second signal in the composite signal  
35     as a function of the comparison.

          69.     The signal conditioner of claim 68 wherein the logic further applies a weighting  
function to a plurality of composite signal samples, estimates the absolute value of the weighted  
samples, determines the maximum of the weighted samples, compares the maximum of the



1 weighted samples to a previous maximum estimation, and selects the maximum of the weighted  
samples as the maximum estimation when the maximum of the weighted samples is greater than  
the previous maximum estimation.

5 70. The signal conditioner of claim 69 wherein the logic applies a scale factor to the  
previous maximum estimation when the estimated maximum of the weighted samples is less than  
the previous maximum estimation, and selects the scaled previous maximum estimation as the  
maximum estimation.

10 71. The signal conditioner of claim 64 further comprising a pre-conditioner applied  
to the composite signal prior to the detection of the second signal thereon.

72. The signal conditioner of claim 71 wherein pre-conditioner comprises a second  
adaptive filter to filter the first signal and a subtractor to subtract the filtered second signal from  
the composite signal.

15 73. A method of synchronizing a data signal in a system having an input clock that  
is uncorrelated to an output clocks, comprising:  
generating an error signal as a function of the difference in the input and output clock; and  
fractionally resampling the data signal as a function of the error signal.

20 74. The method of claim 73 wherein the step of generating an error signal comprises  
driving a first counter with the input clock, driving a second counter with the output clock  
wherein said error signal is generated by dividing the output of the first counter by the output of  
the second counter.

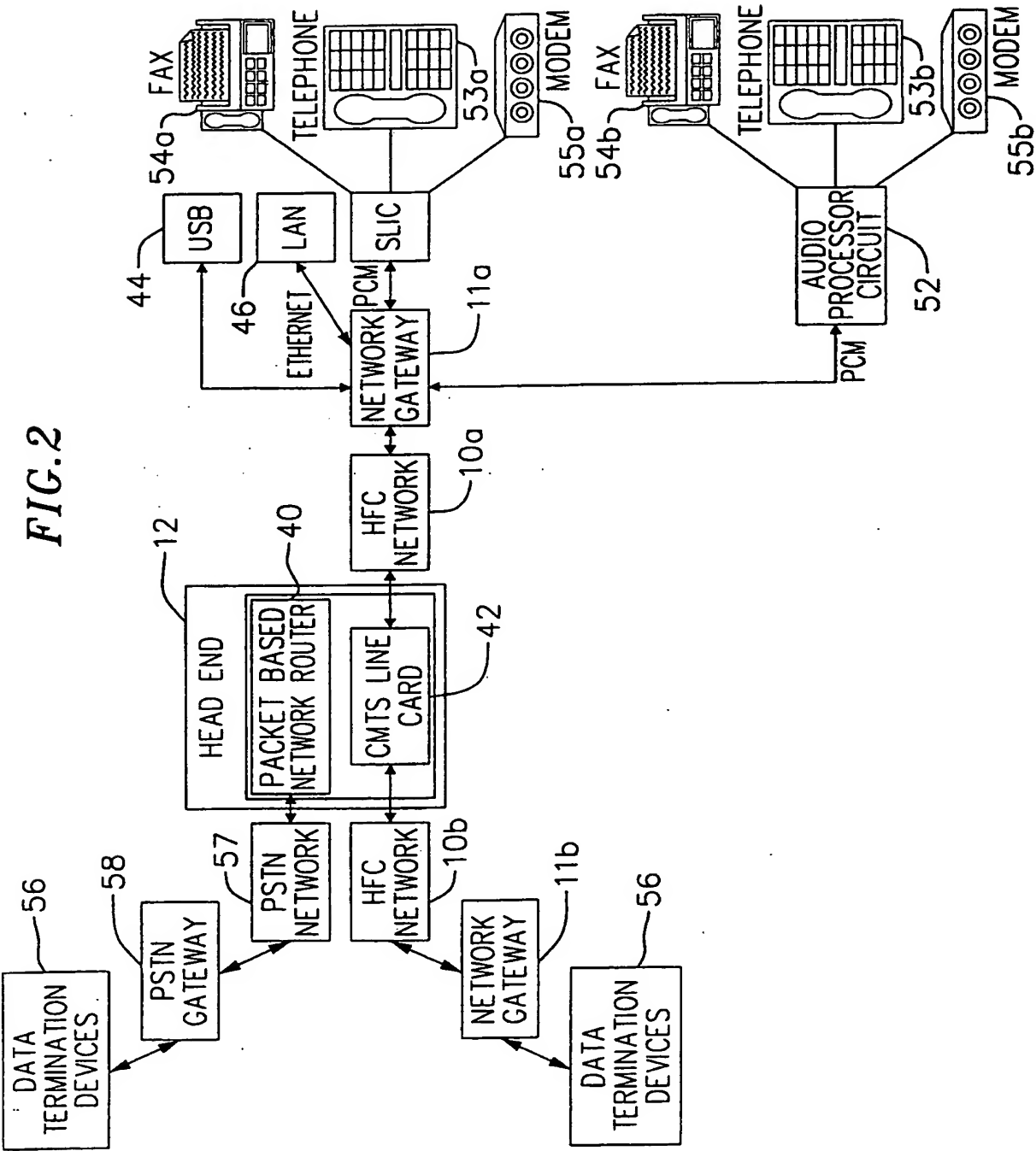
25 75. The method of claim 74 wherein the data signal is fractionally up sampled by the  
error signal when the output of the first counter exceeds the output of the second counter.

76. The method of claim 74 wherein the data signal is fractionally down sampled by  
the error signal when the output of the first counter is less than the output of the second counter.

30 77. The method of claim 74 further comprising low pass filtering the outputs of said  
first and second clocks wherein said error signal is generated by dividing the filtered output of  
the first counter by the filtered output of the second counter.

35 78. The method of claim 73 wherein the step of generating an error signal comprises  
driving a first counter with the input clock, clocking the counter with a third clock that is  
correlated with the output clock, wherein said error signal





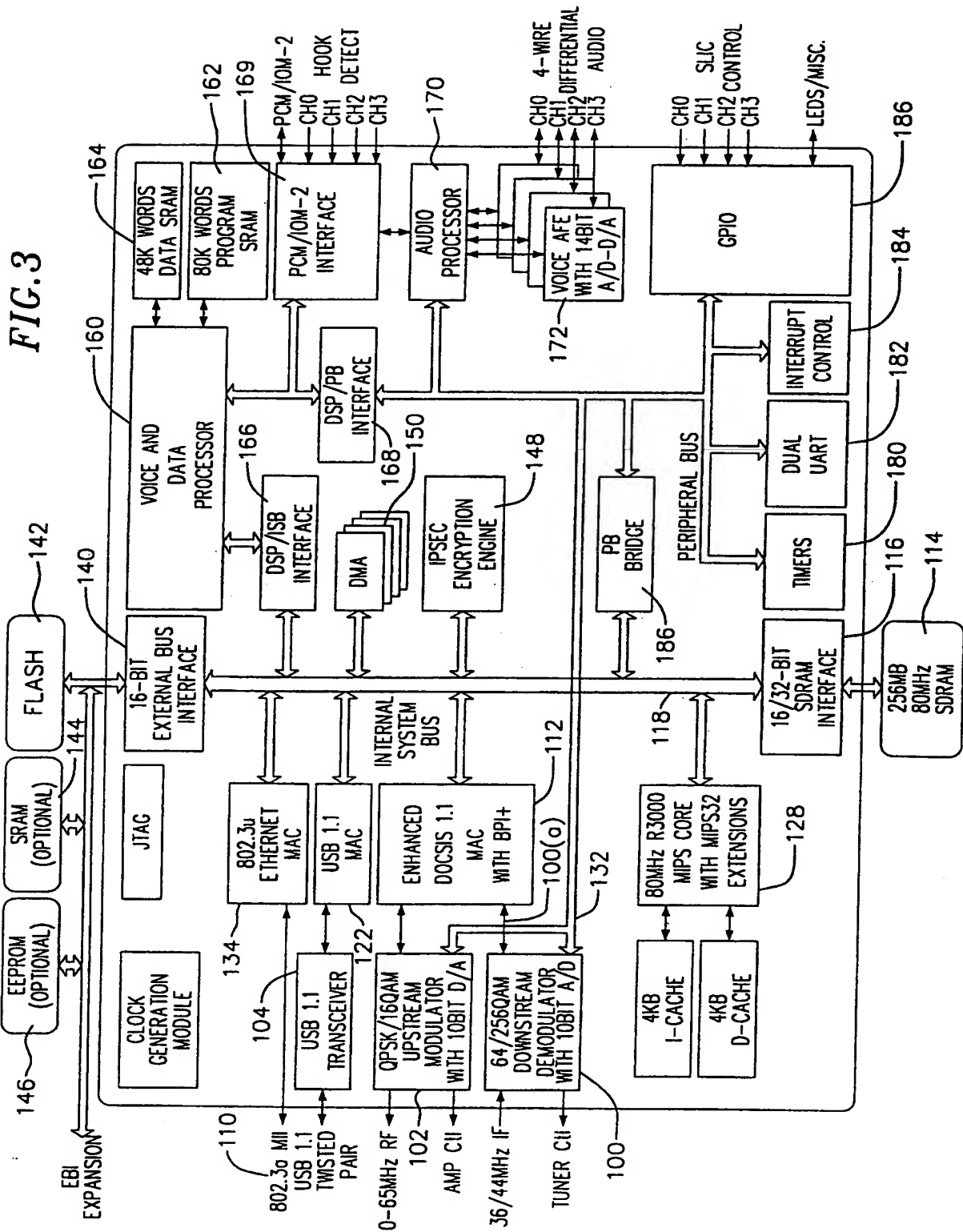
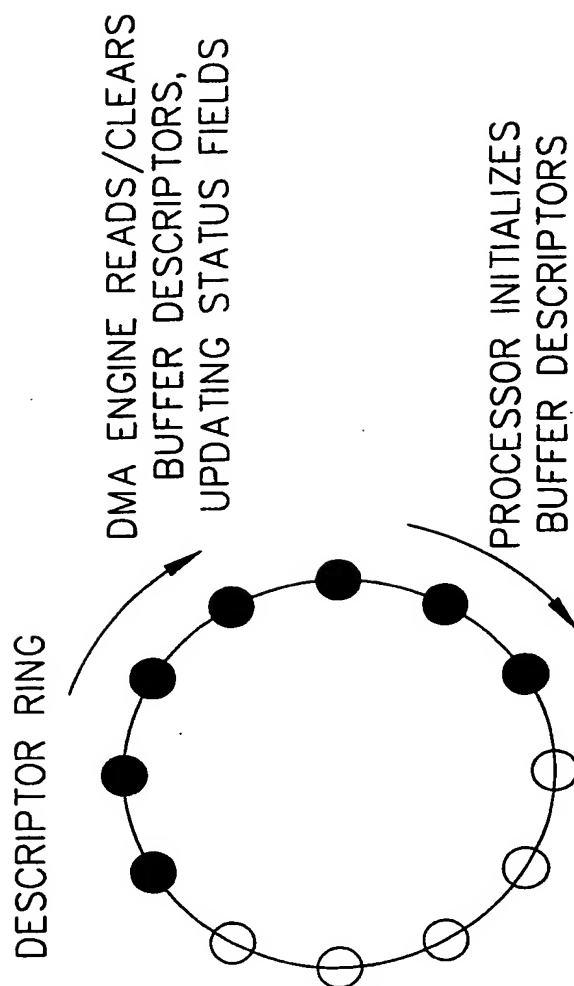


FIG. 4



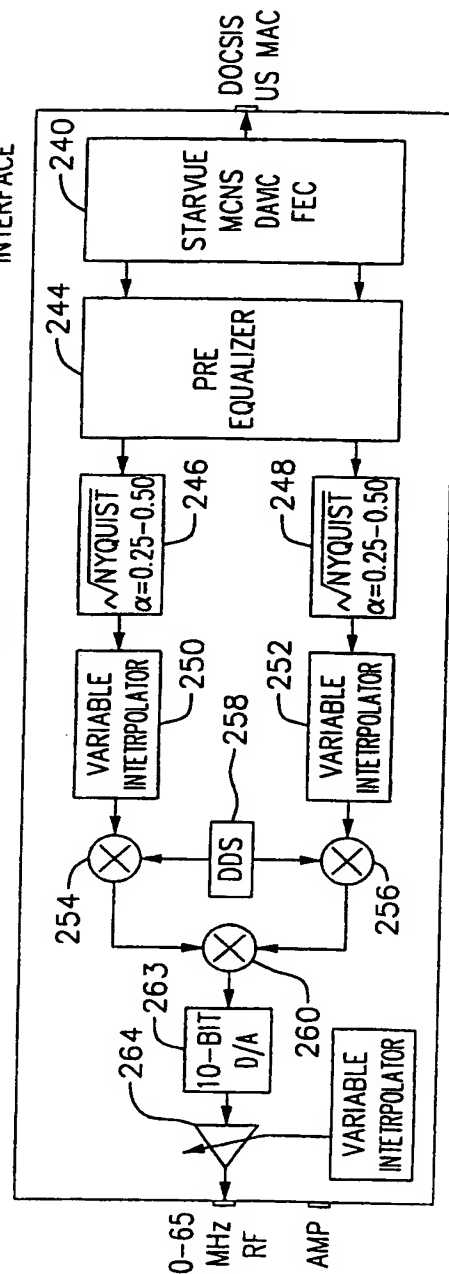
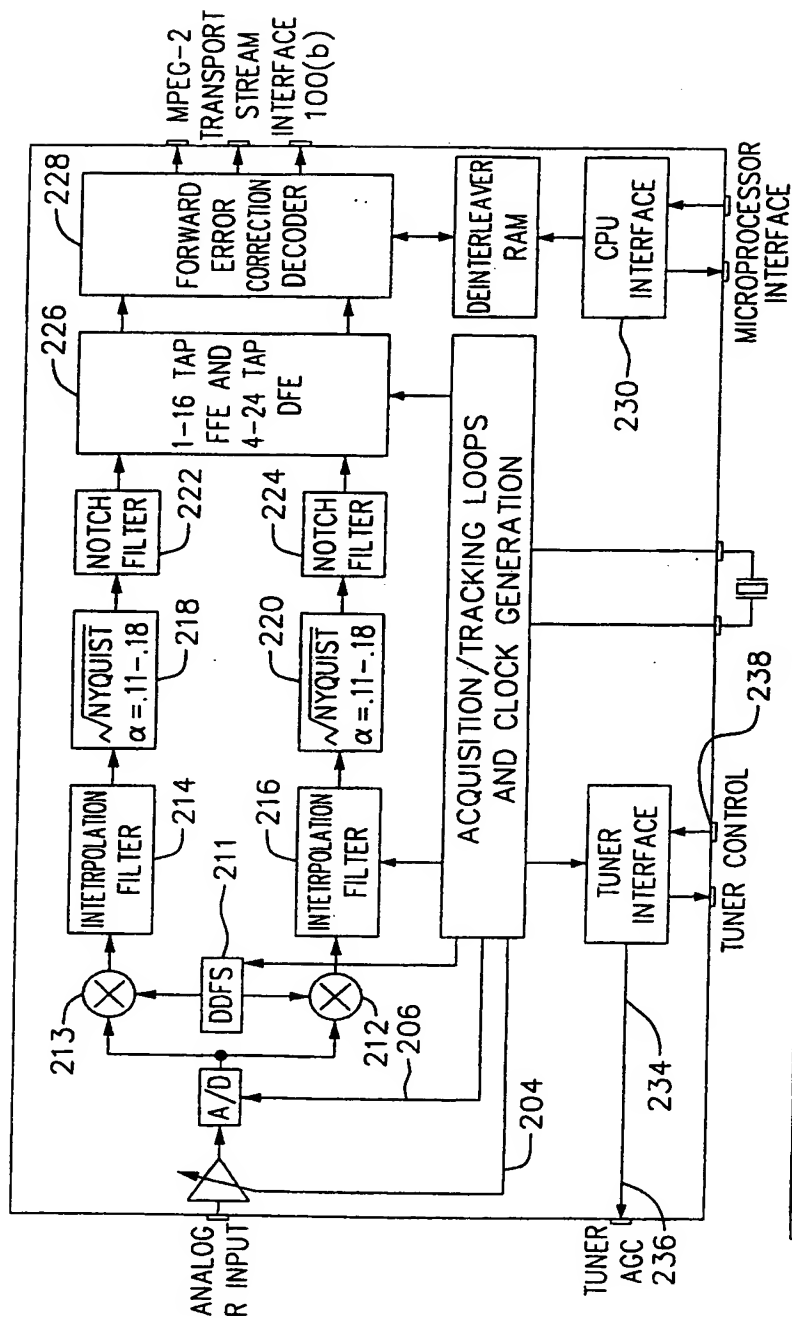


FIG. 7

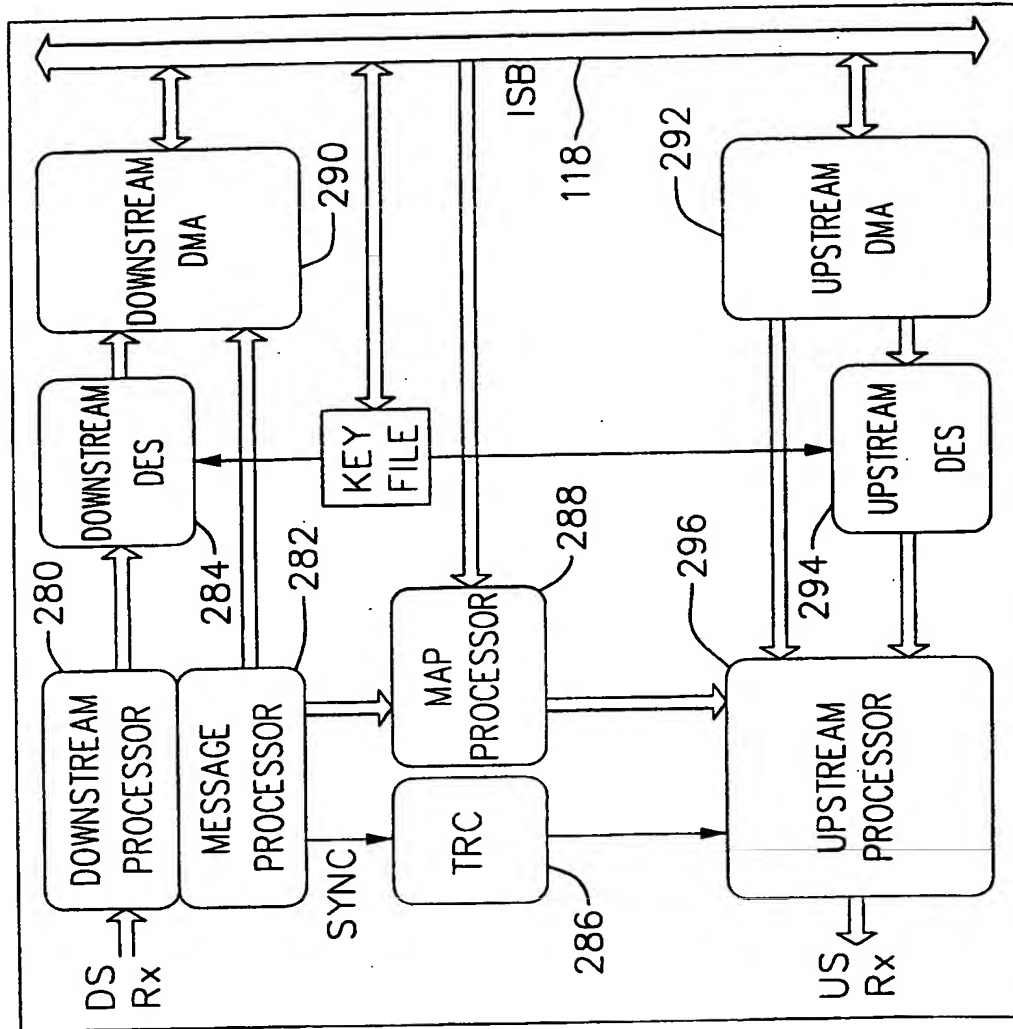


FIG. 8

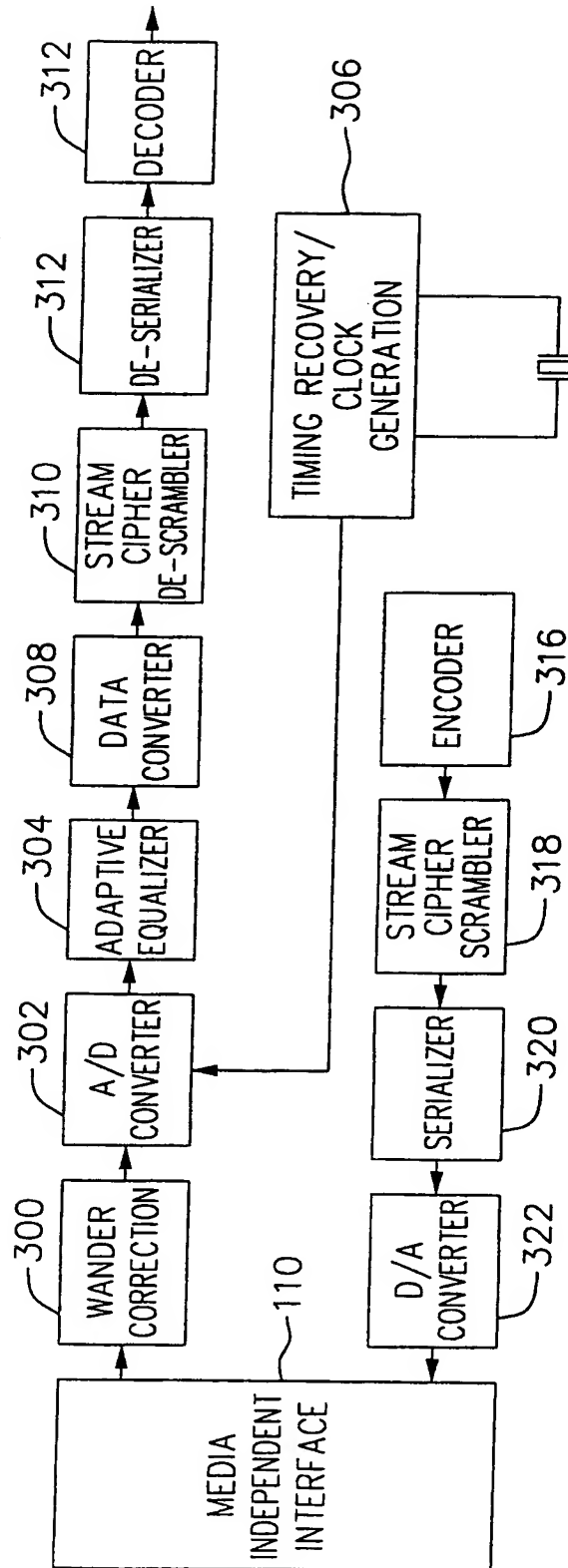
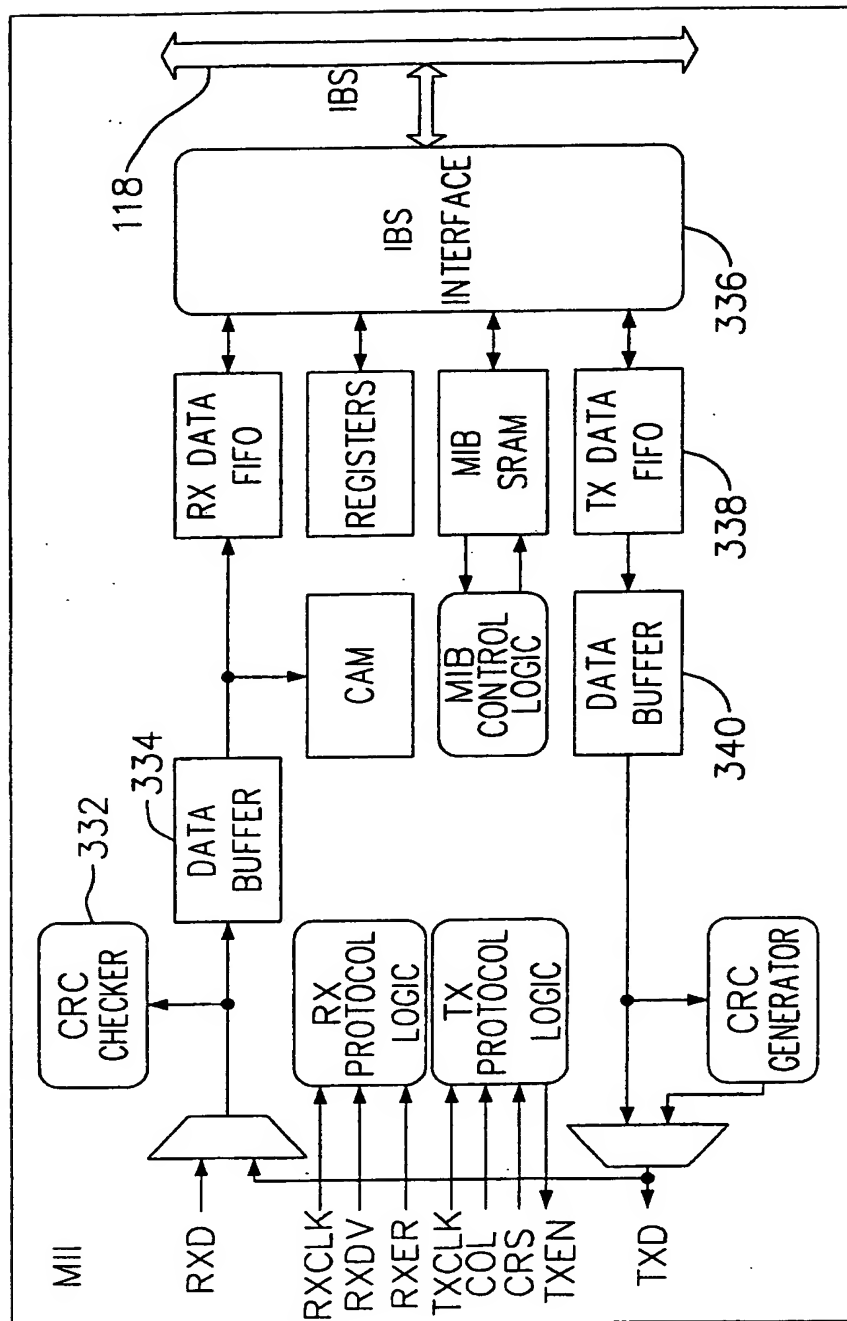




FIG. 9



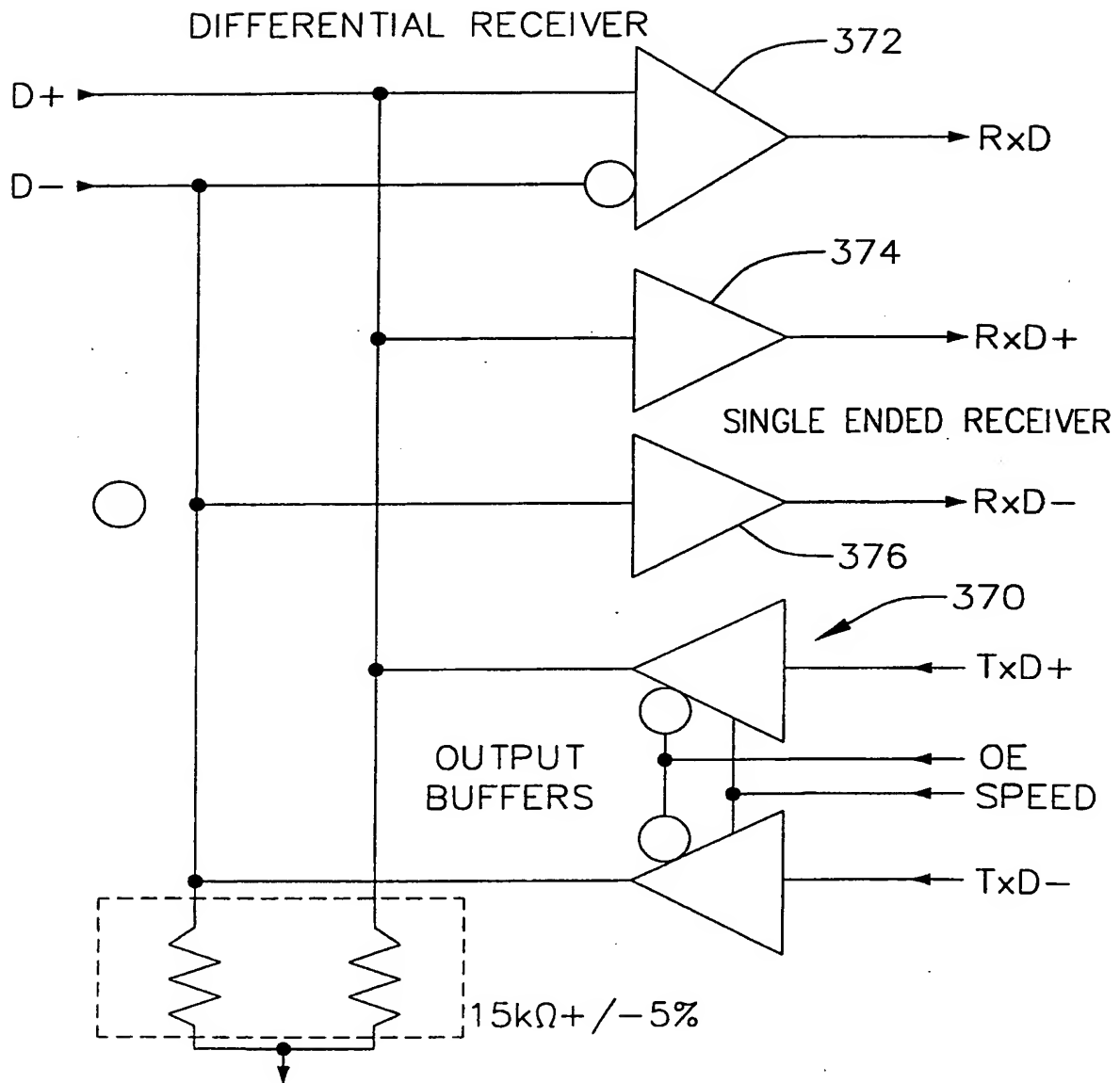
*FIG. 10*

FIG. 10A

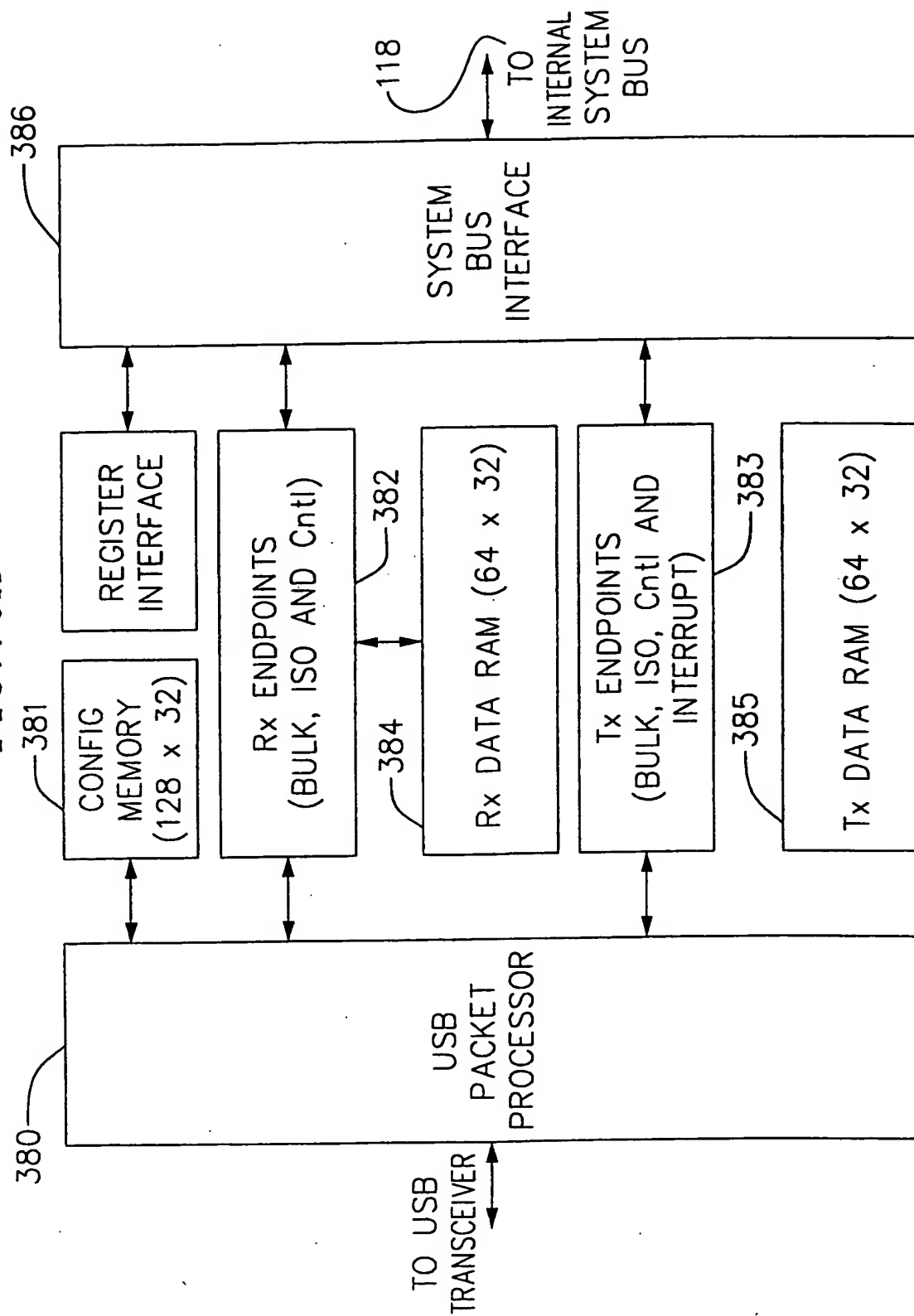


FIG. 11

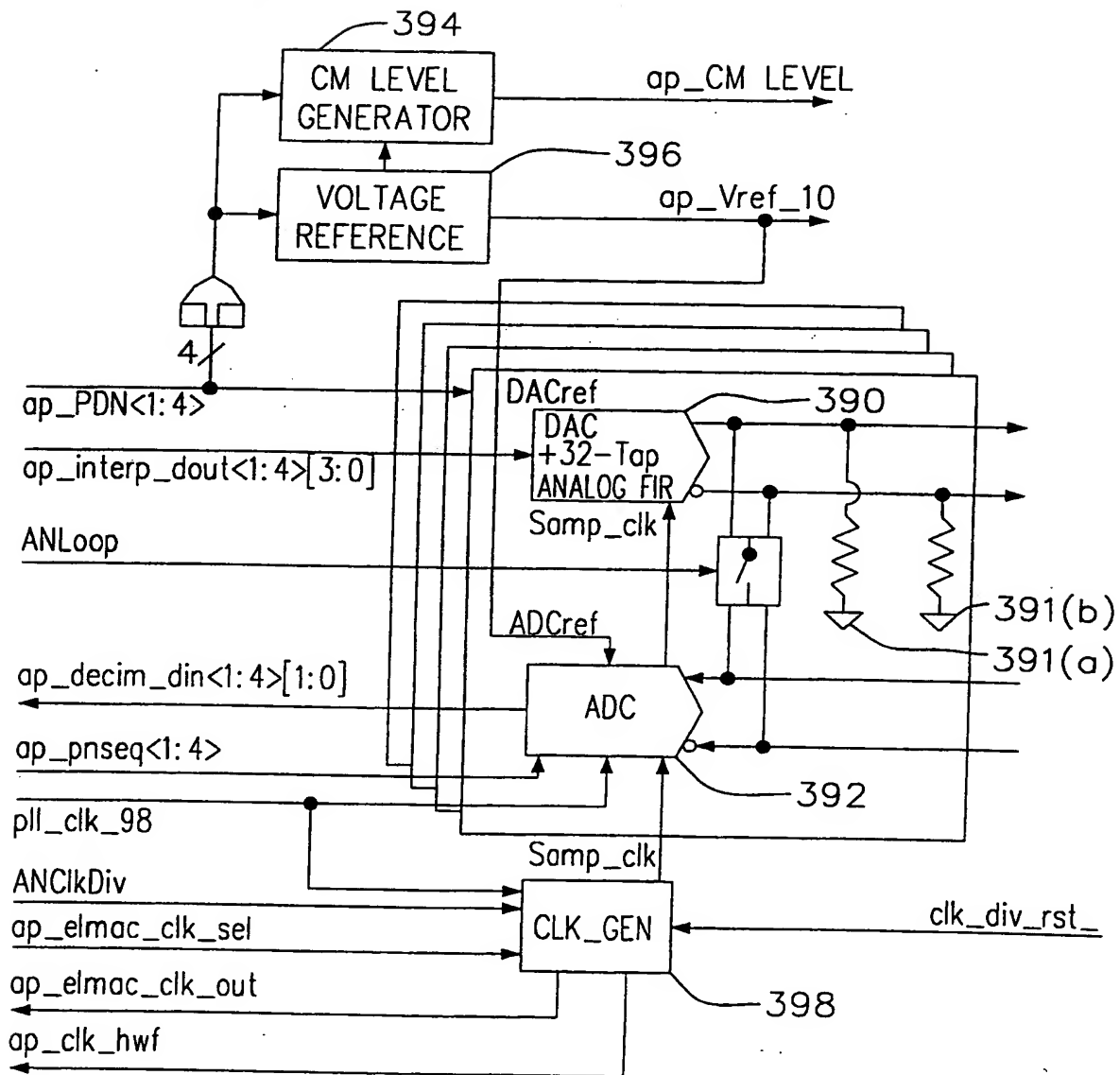




FIG. 12

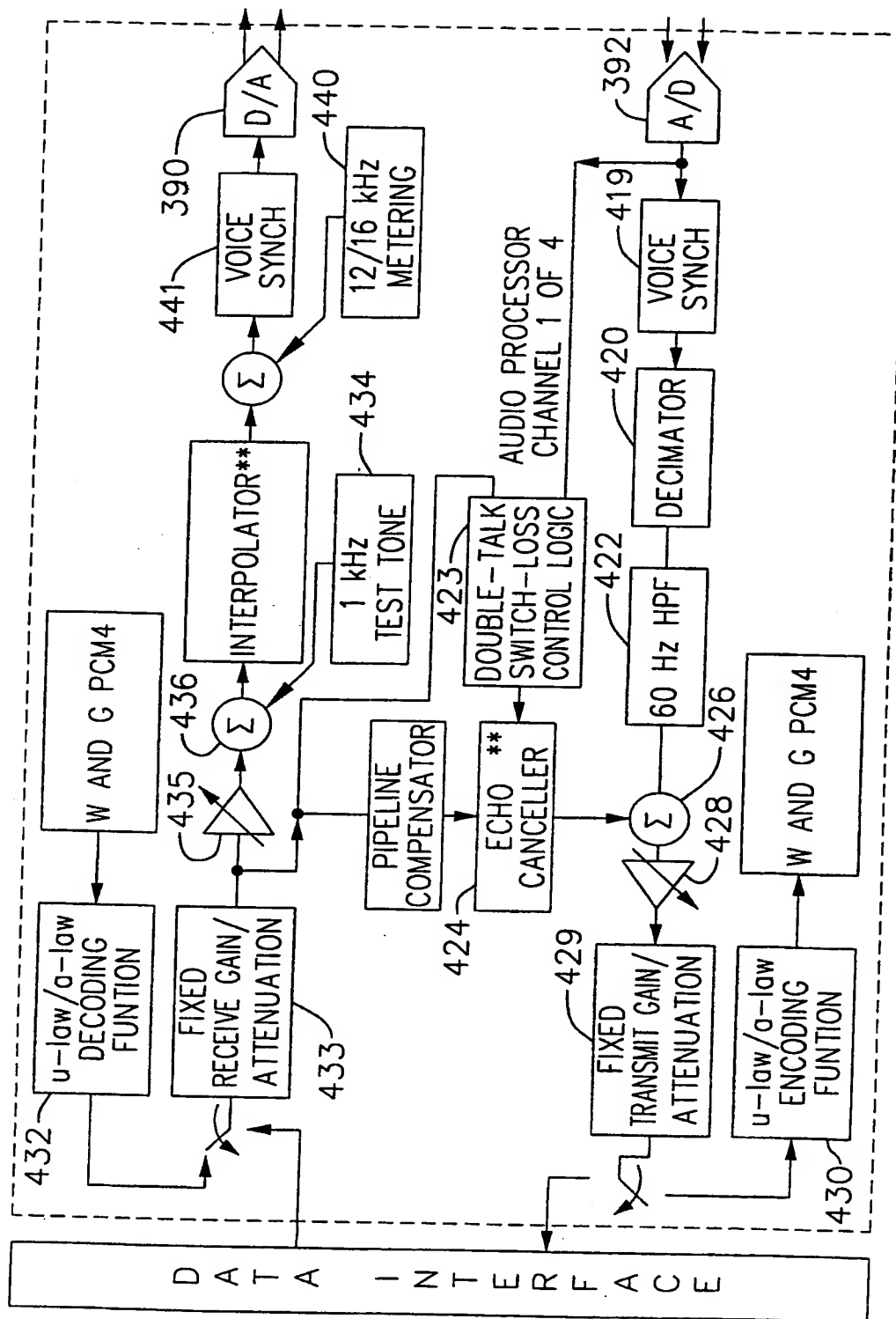
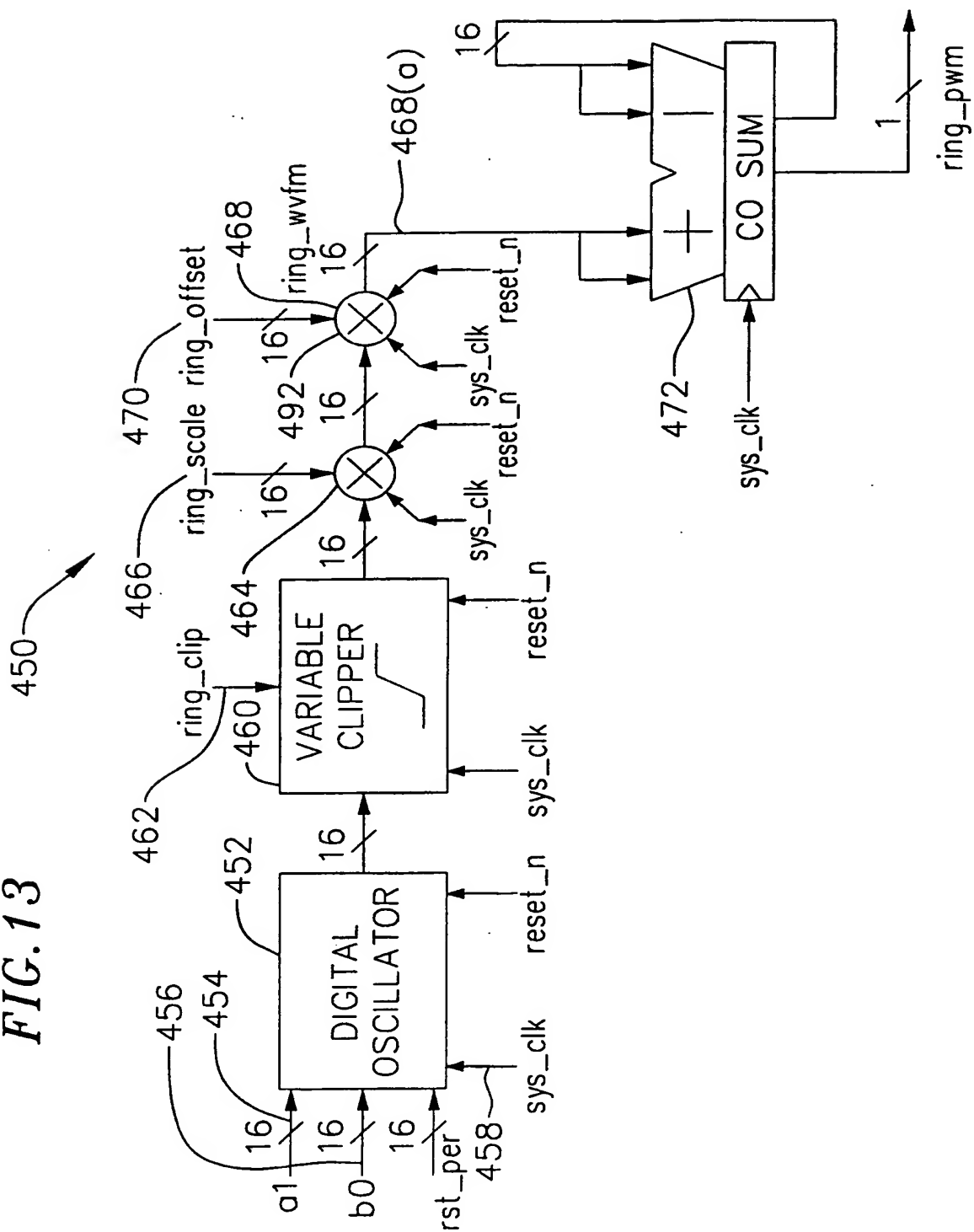


FIG. 13



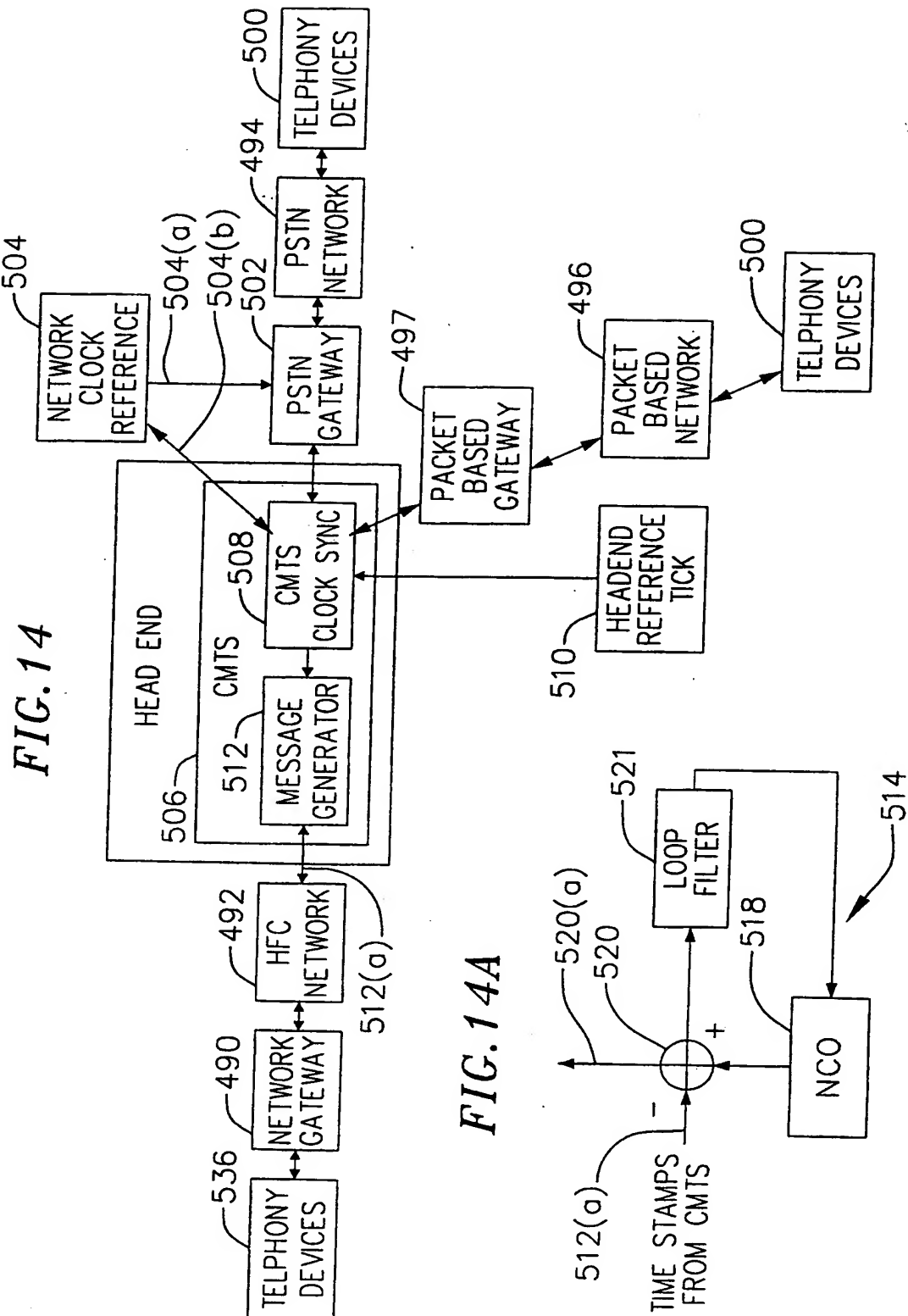




FIG. 15

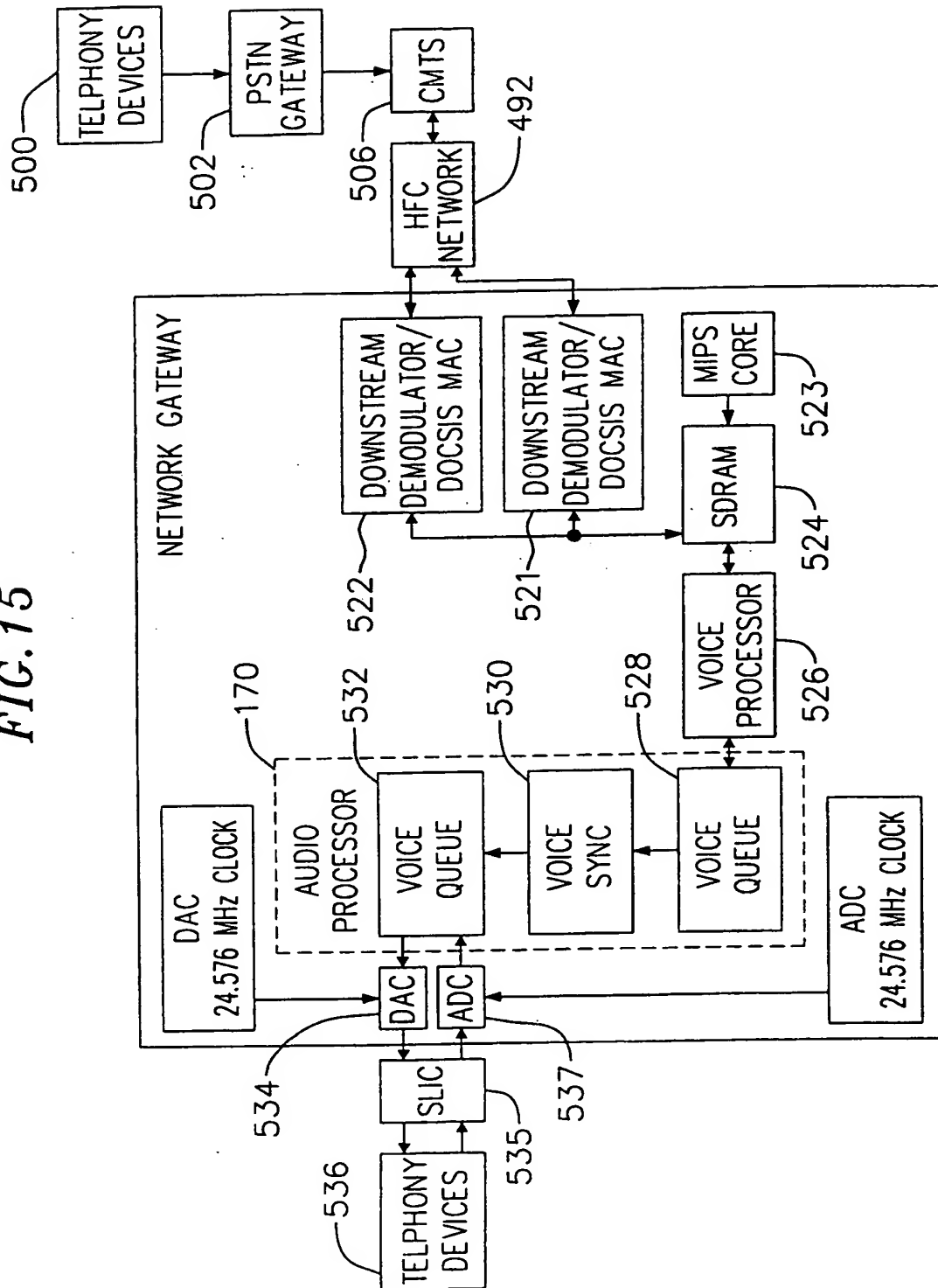


FIG. 16

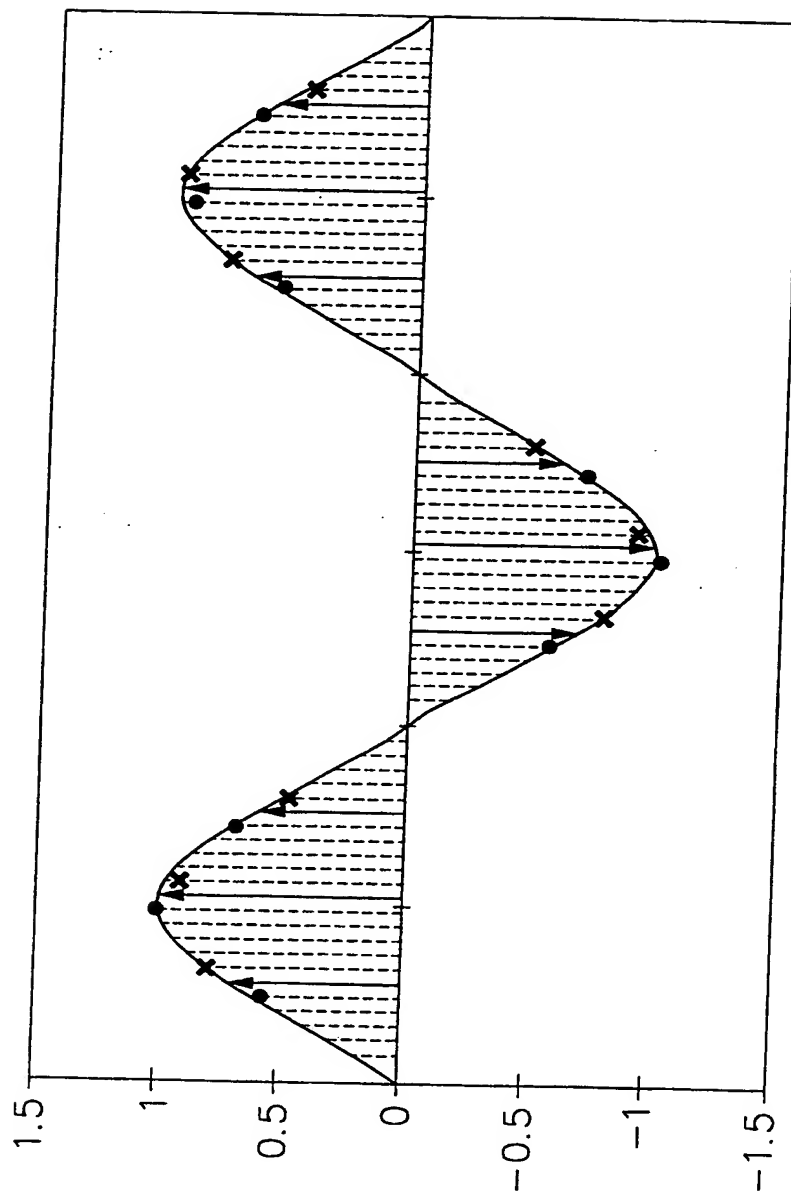


FIG. 17

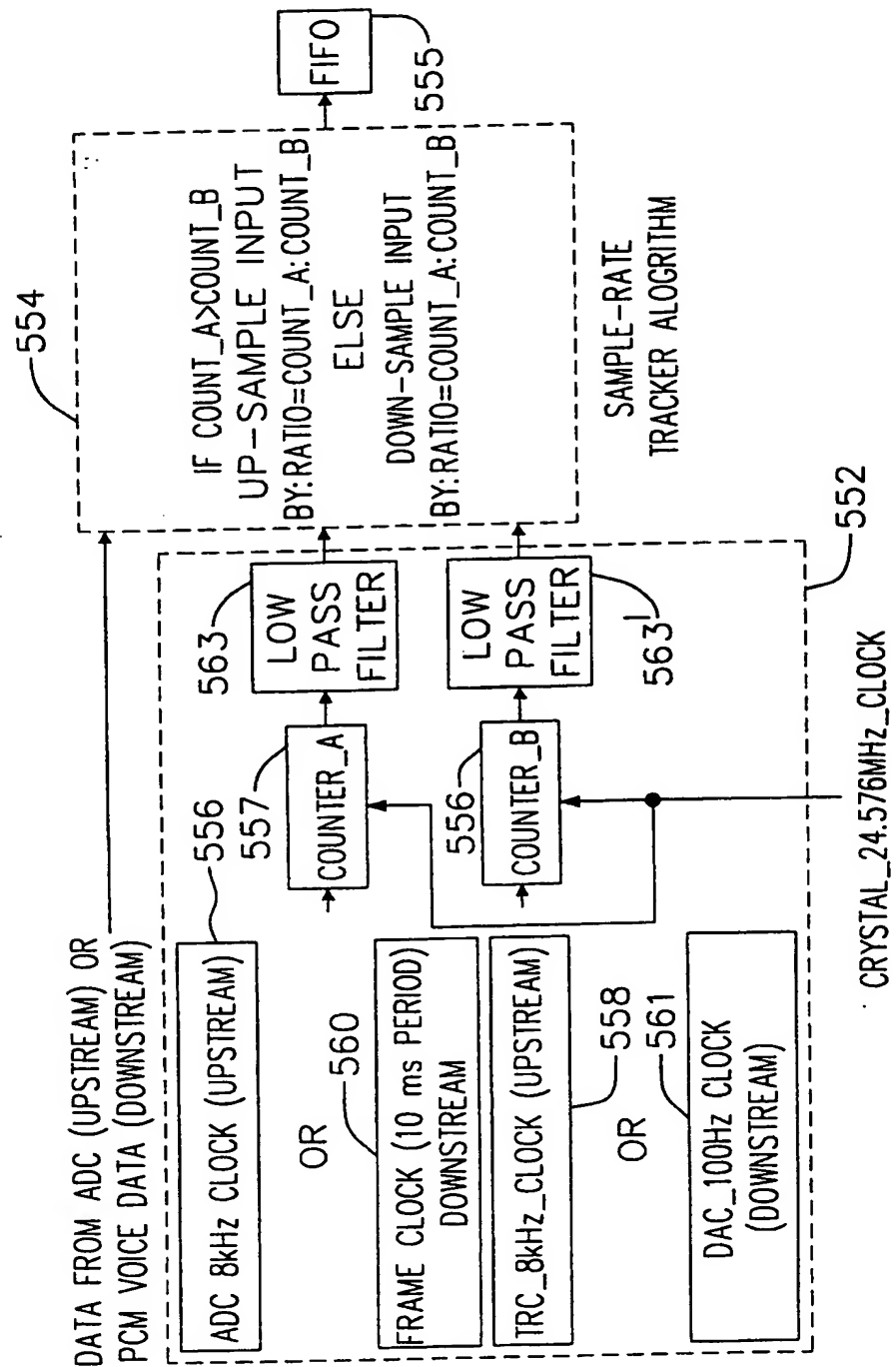
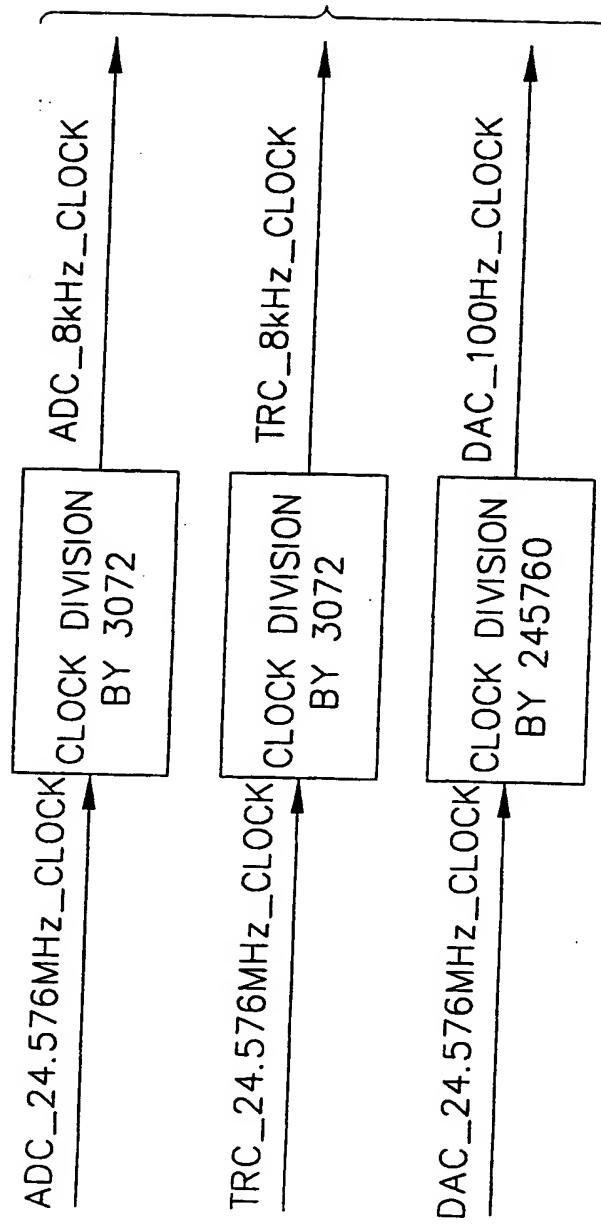
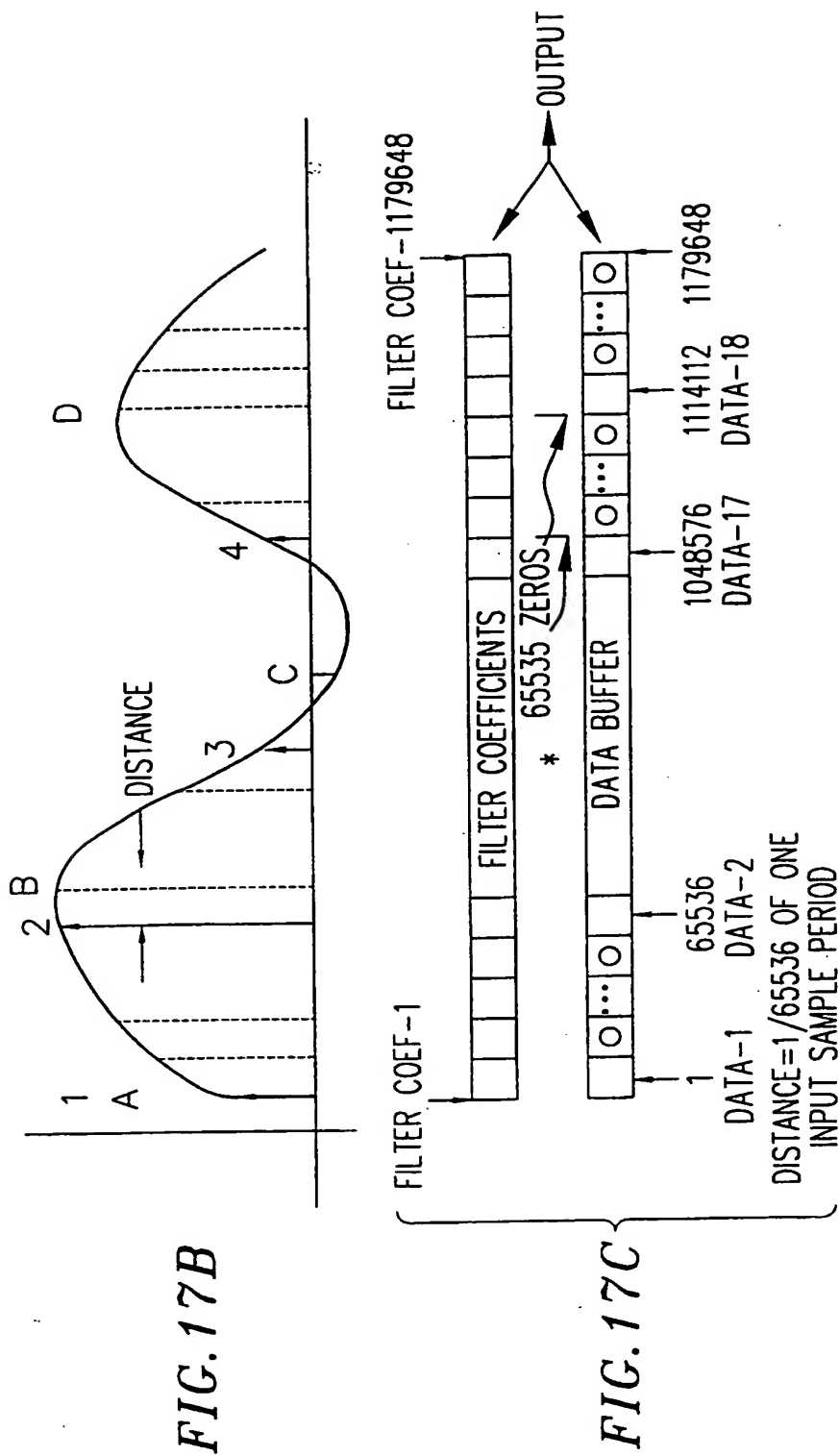


FIG. 17A





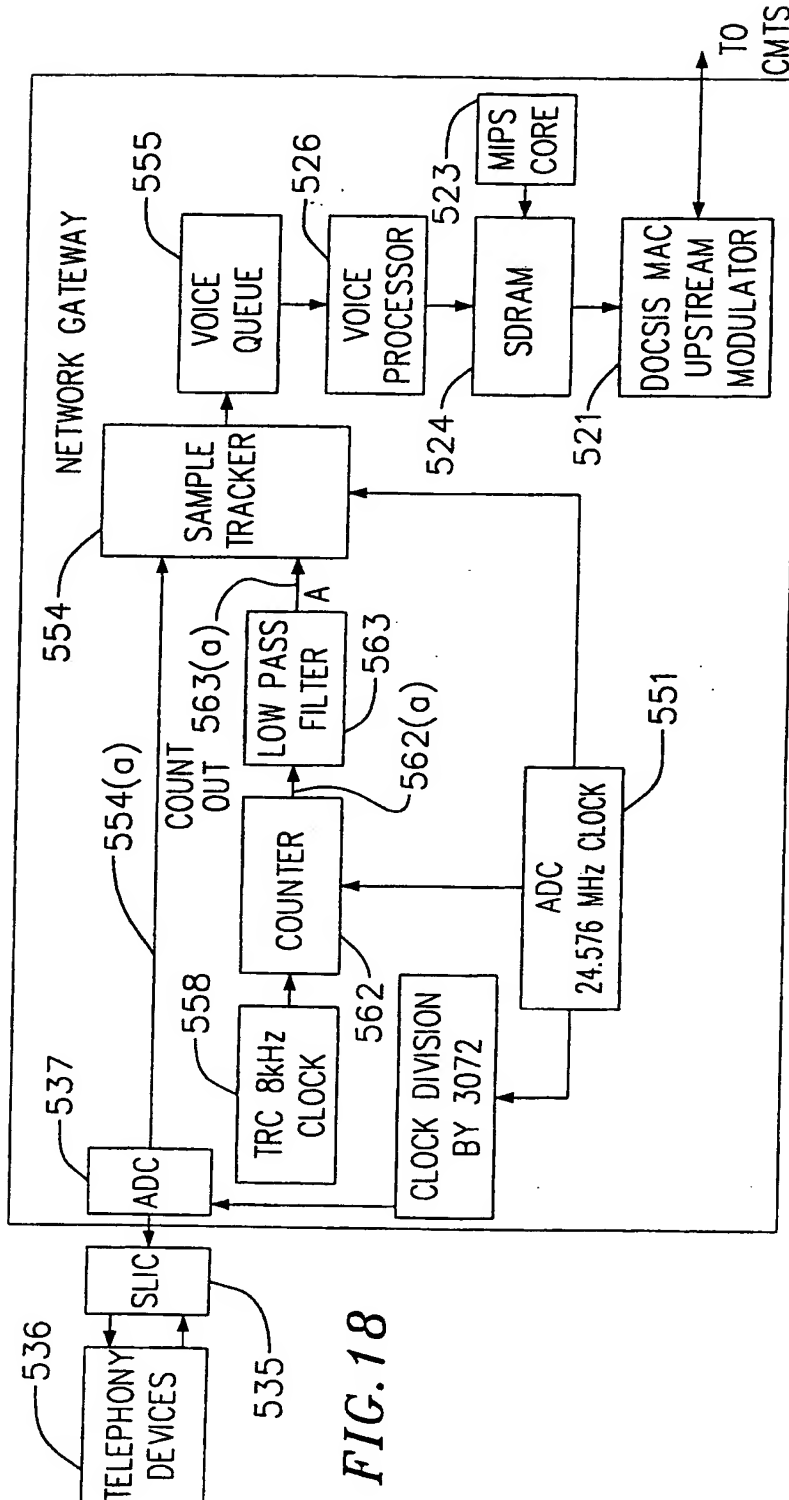


FIG. 18

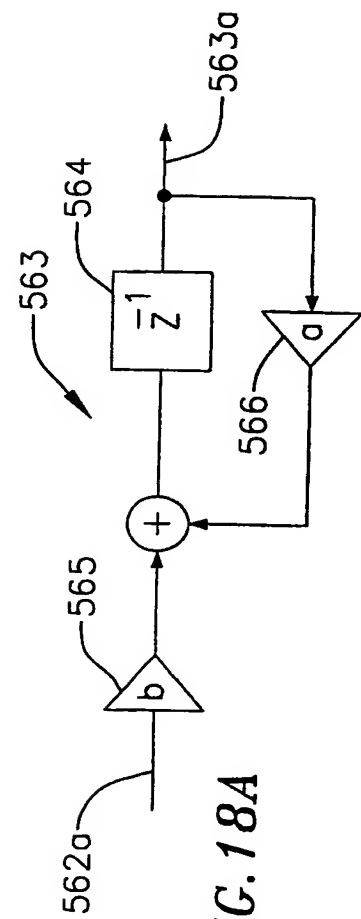


FIG. 18A

FIG. 18B

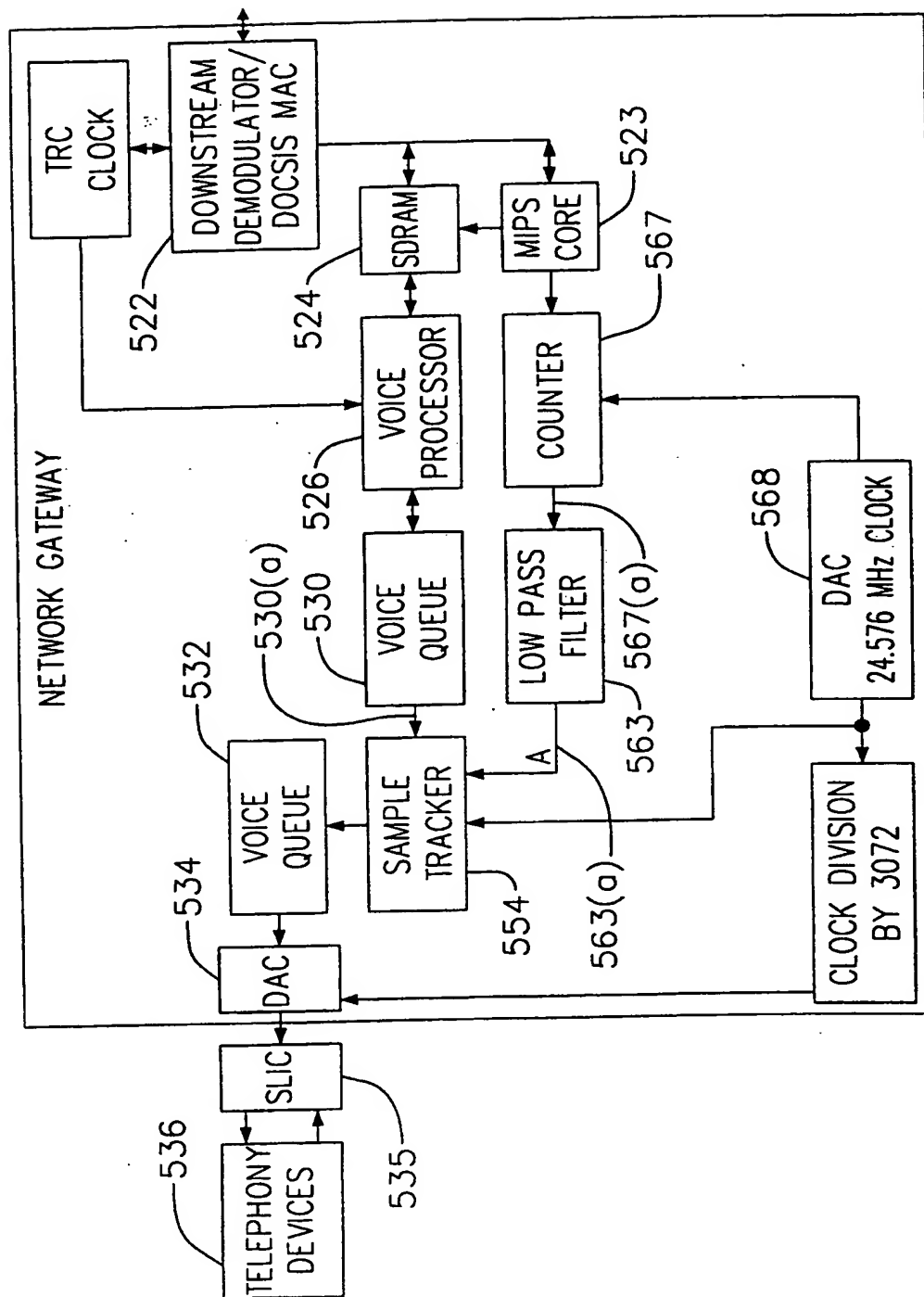


FIG. 19

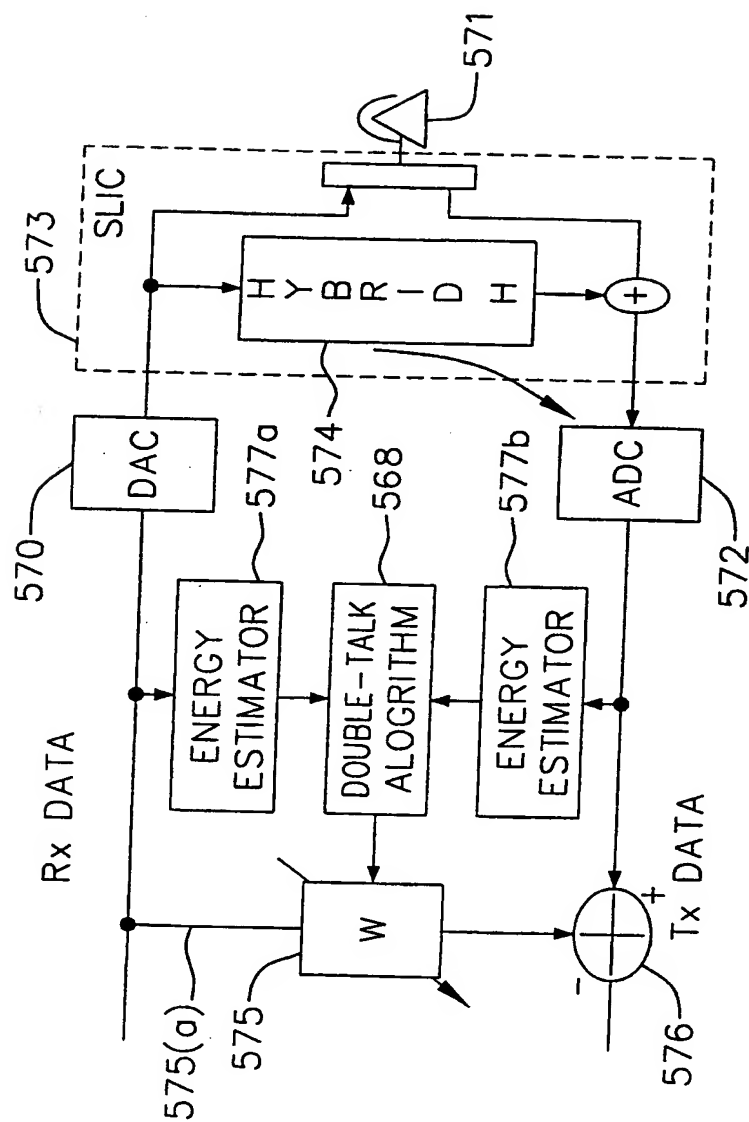
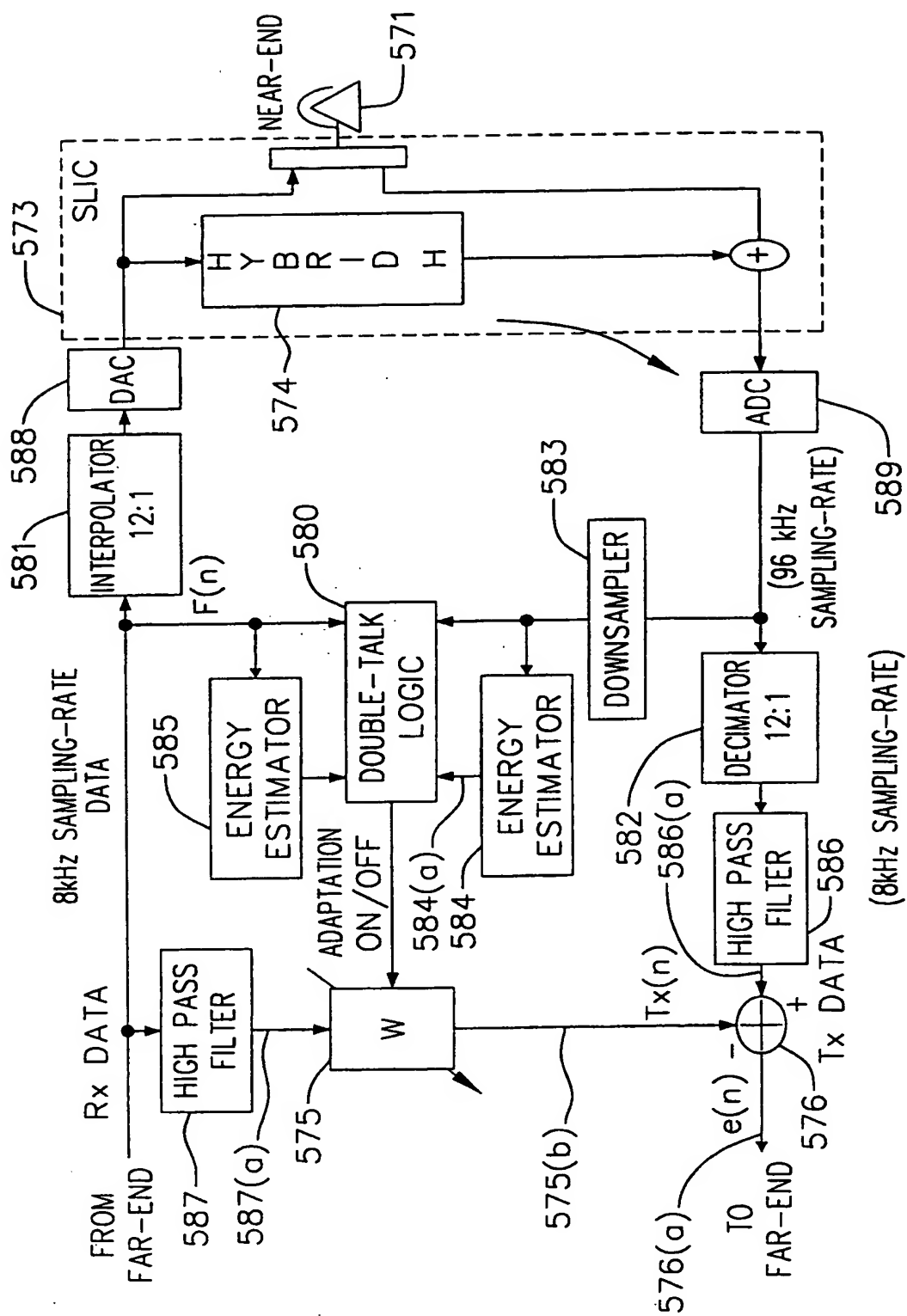




FIG. 20





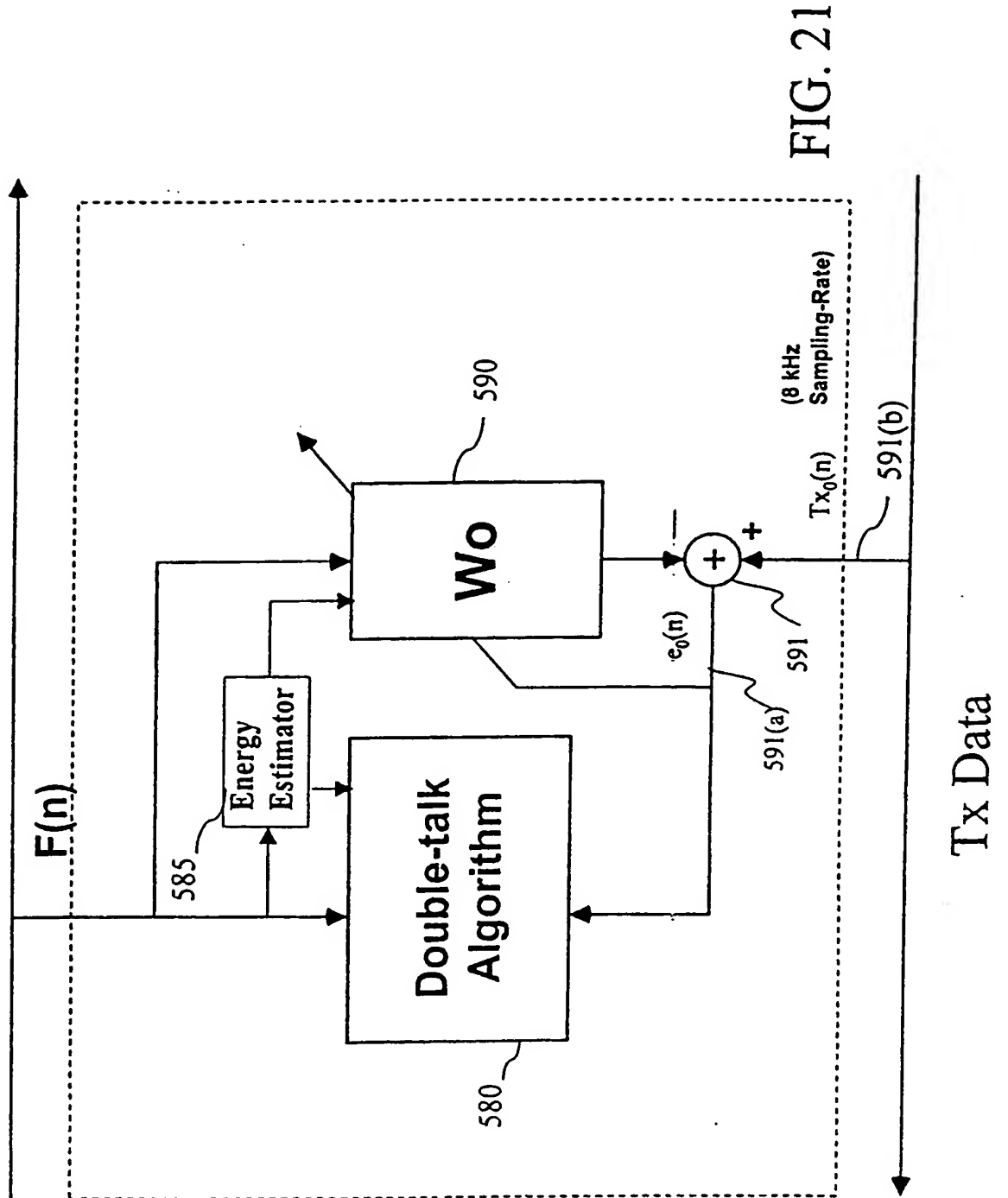
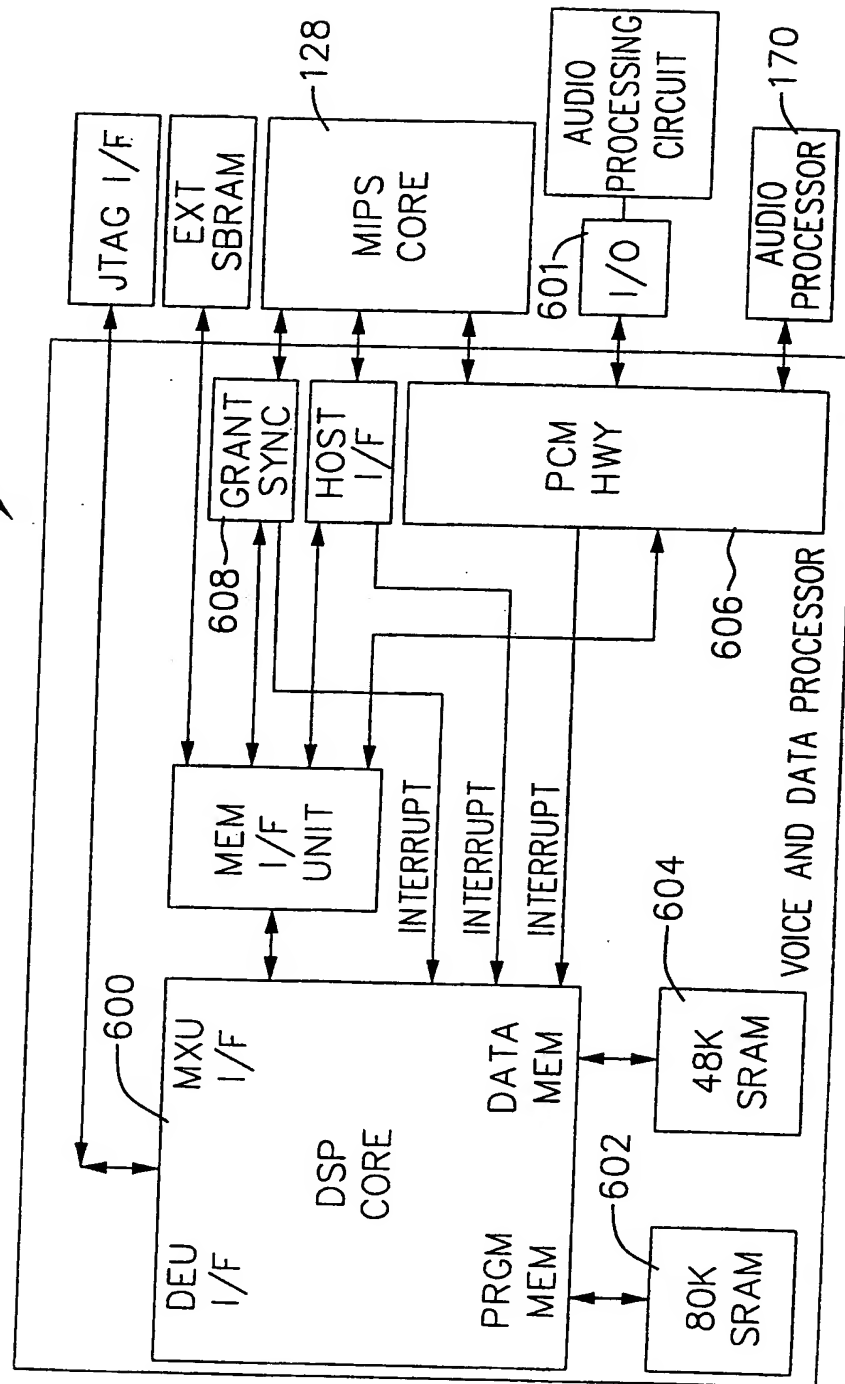


FIG. 22



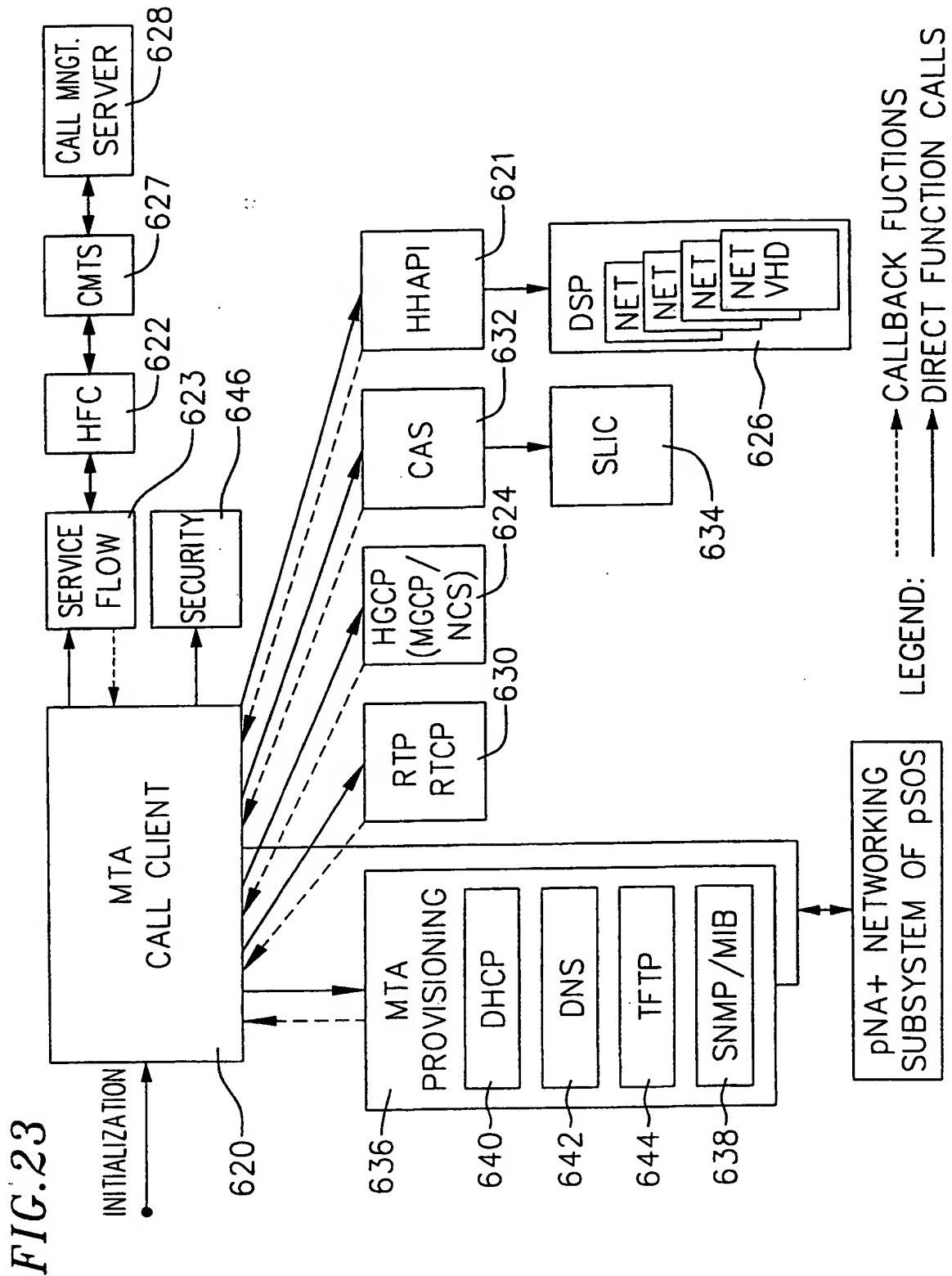


FIG. 24

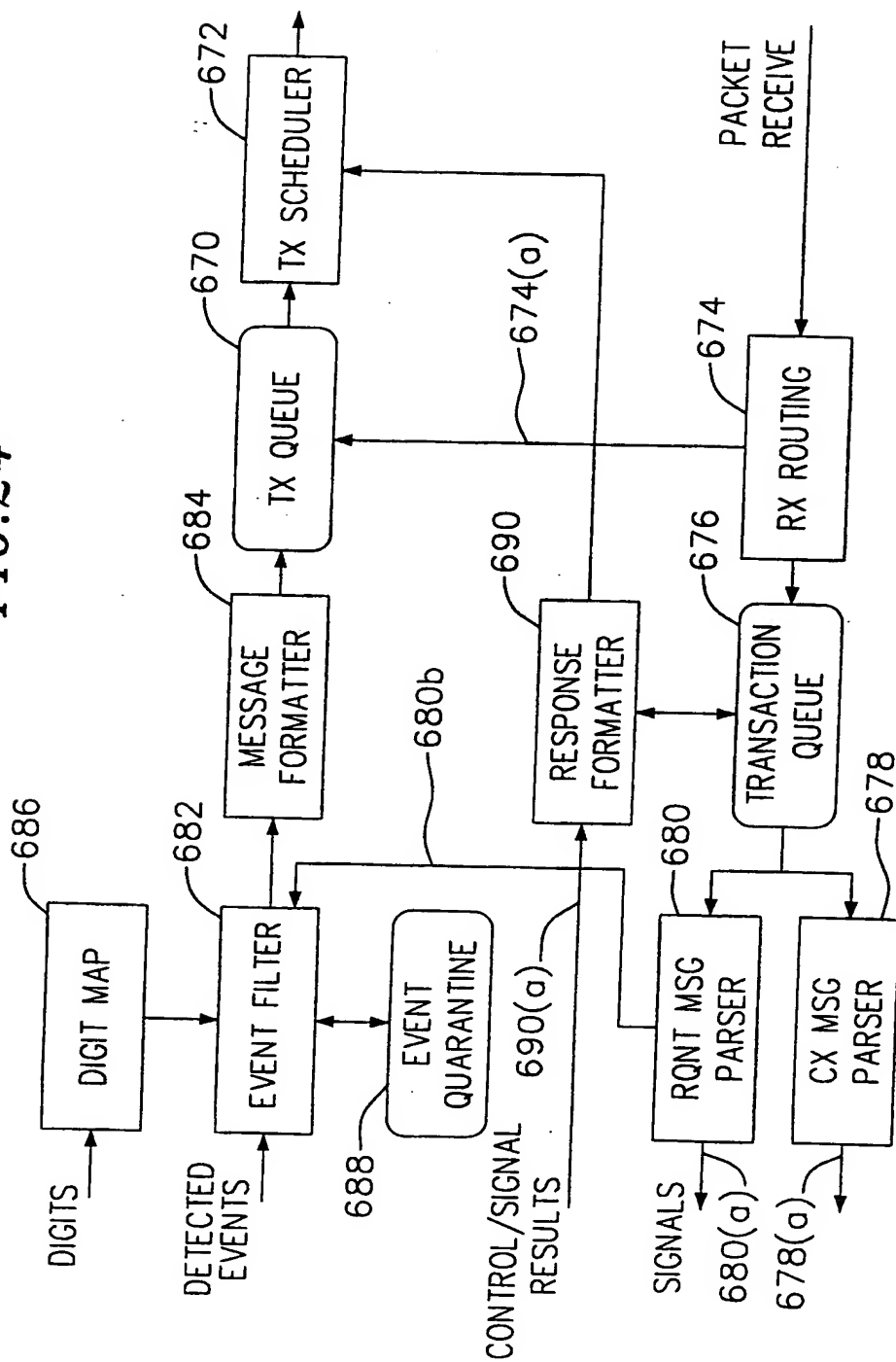
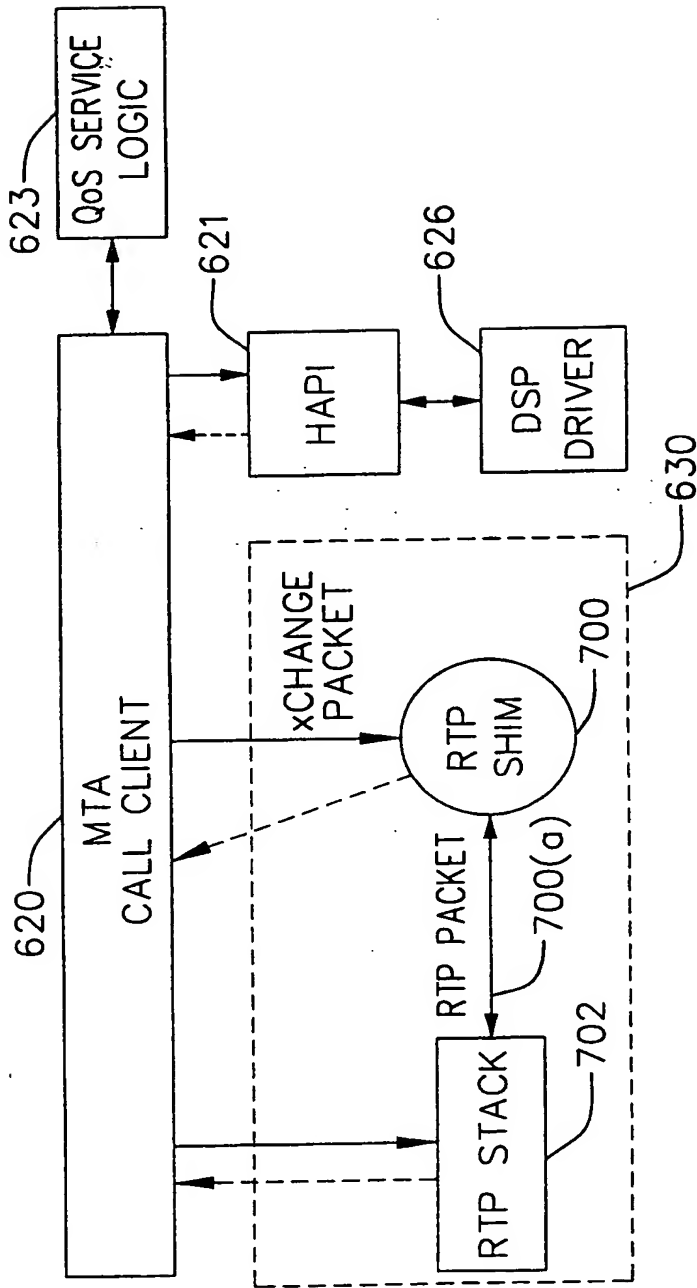


FIG. 25



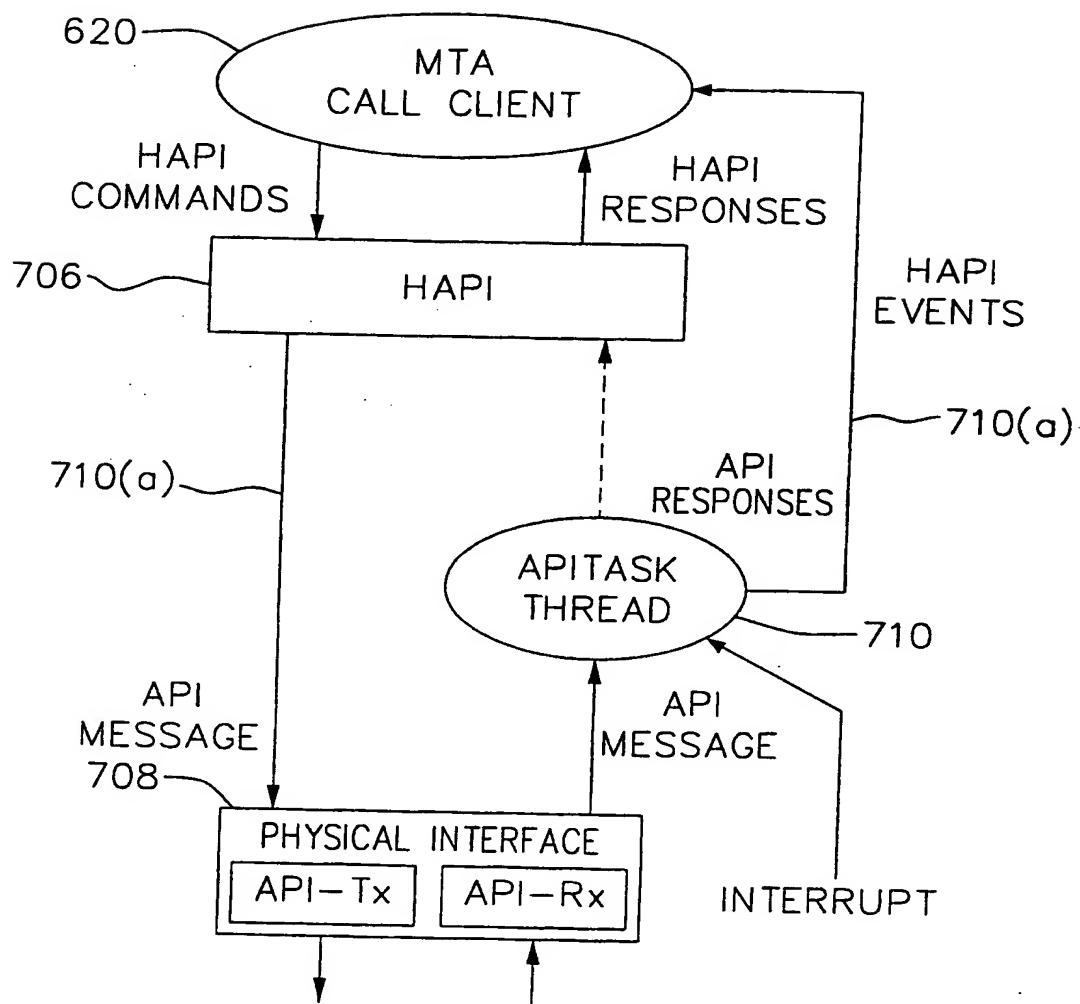
*FIG. 26*



FIG. 27

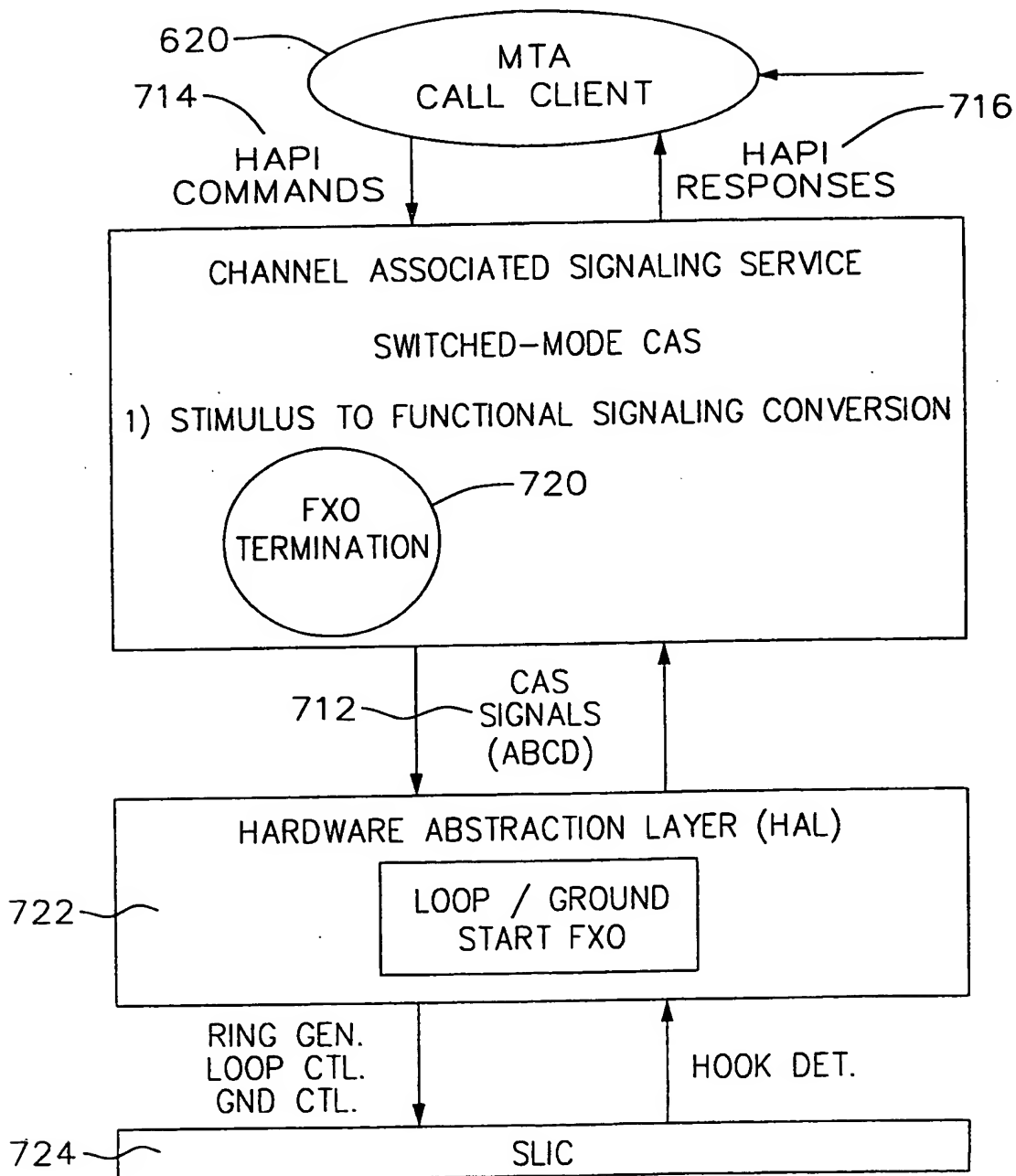


FIG. 28

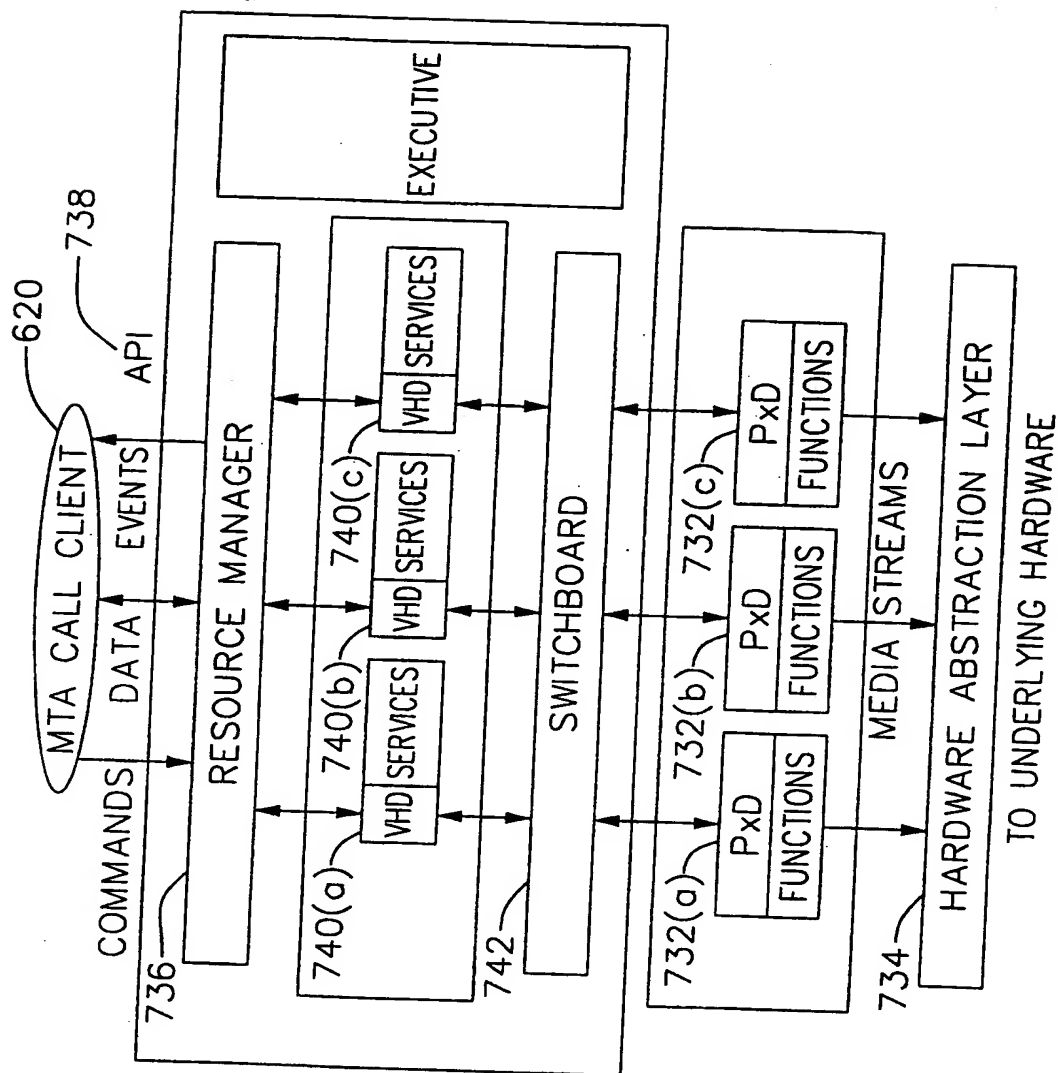


FIG. 29

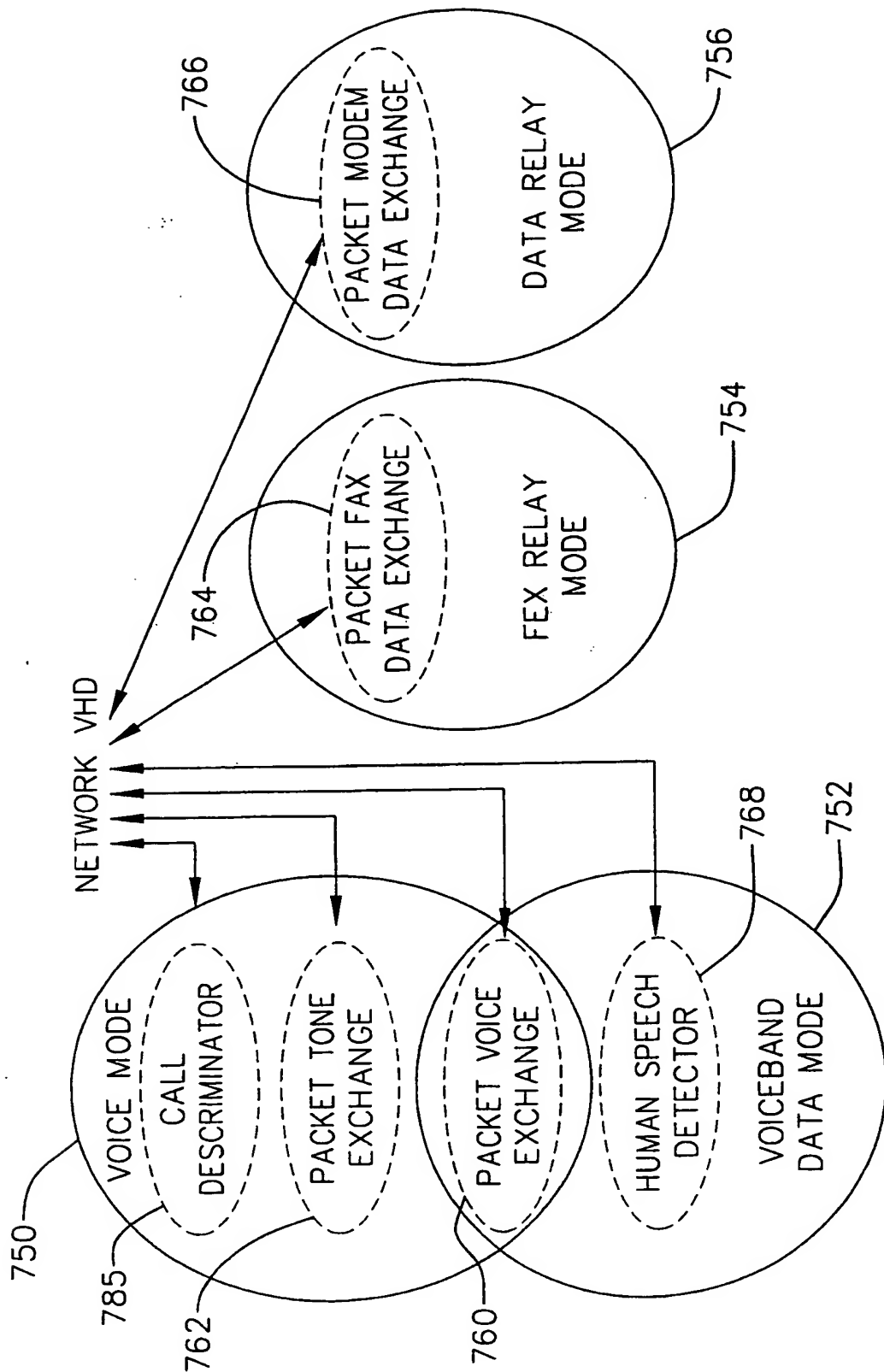
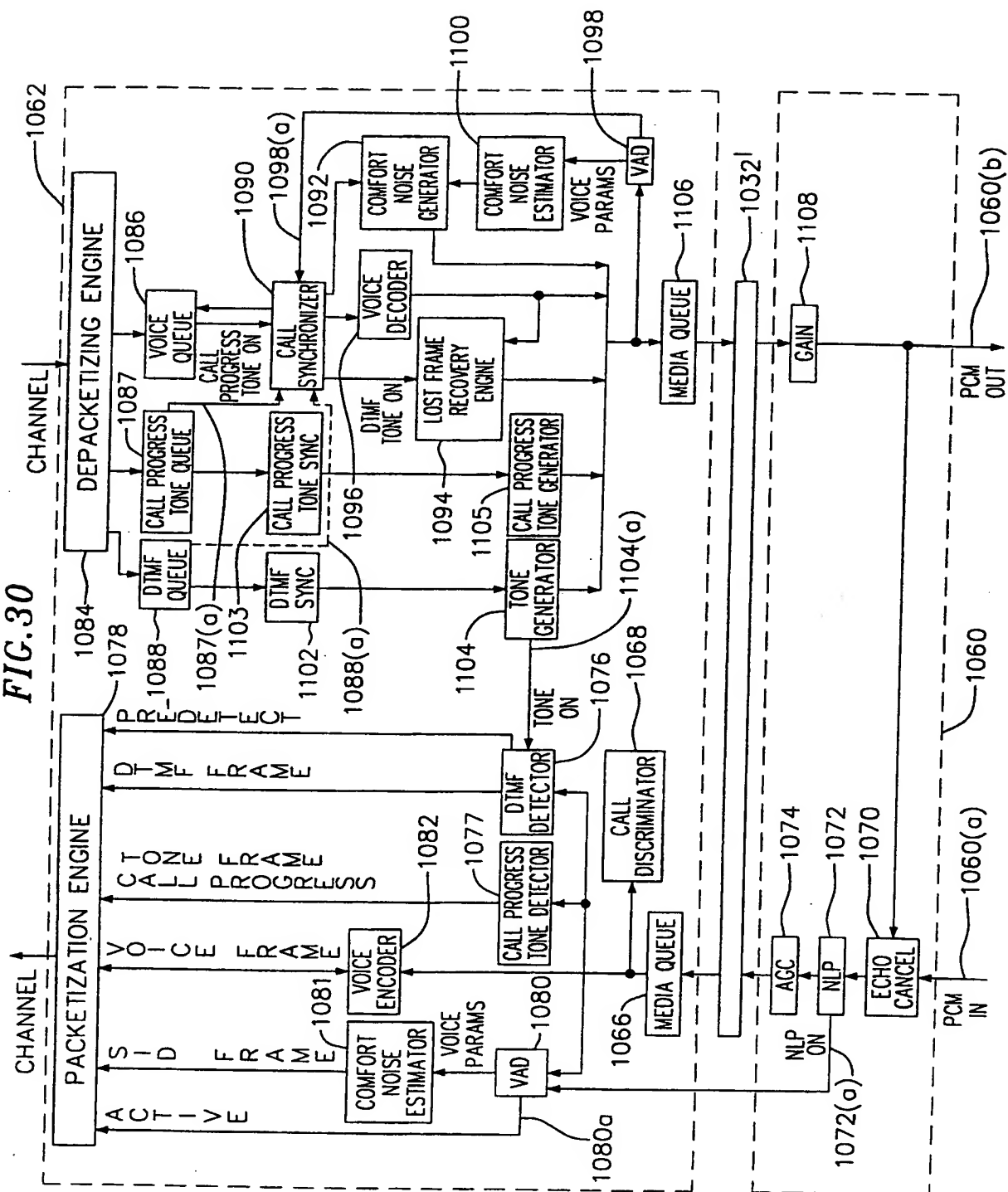
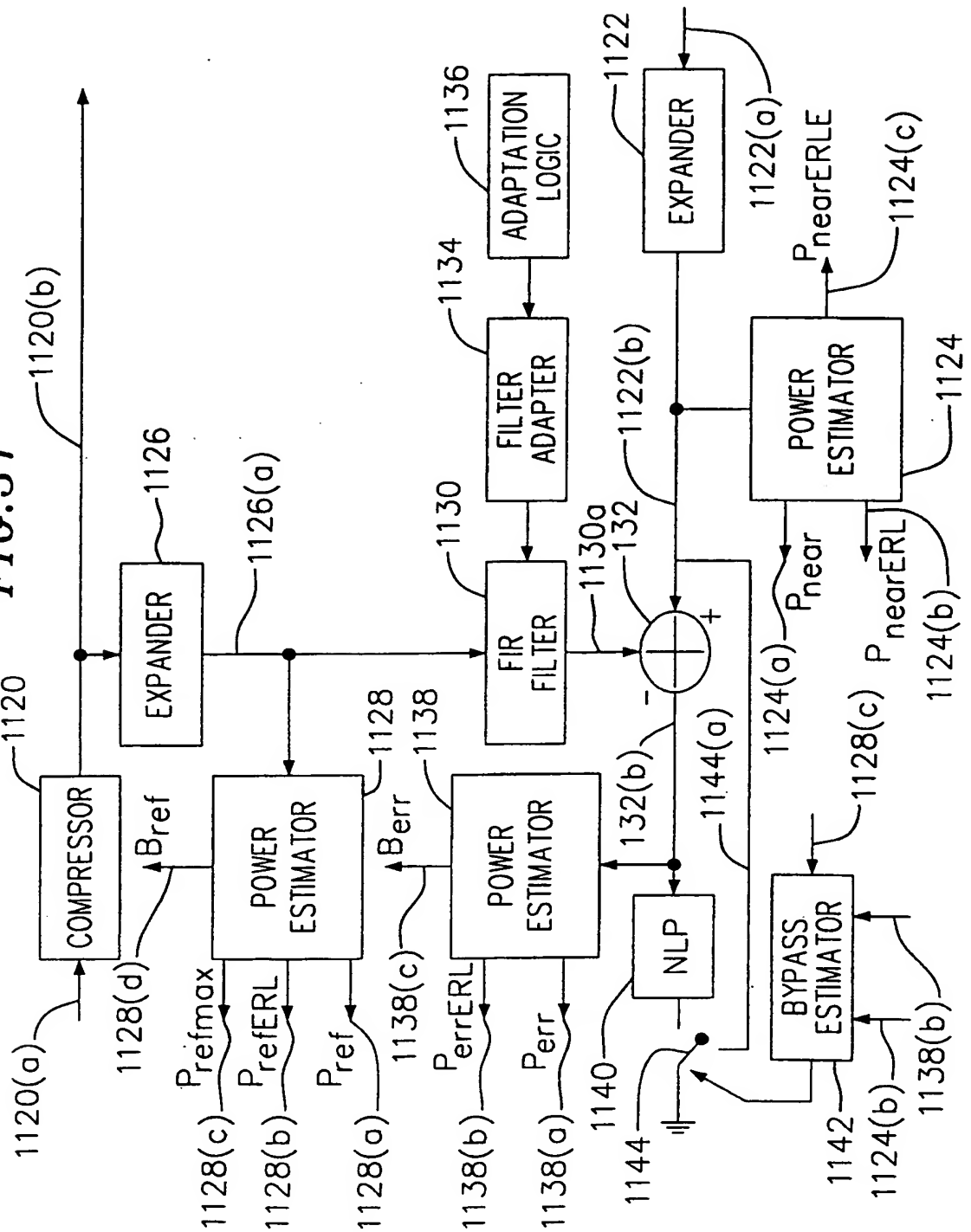


FIG. 30



**SUBSTITUTE SHEET (RULE 26)**

FIG. 31



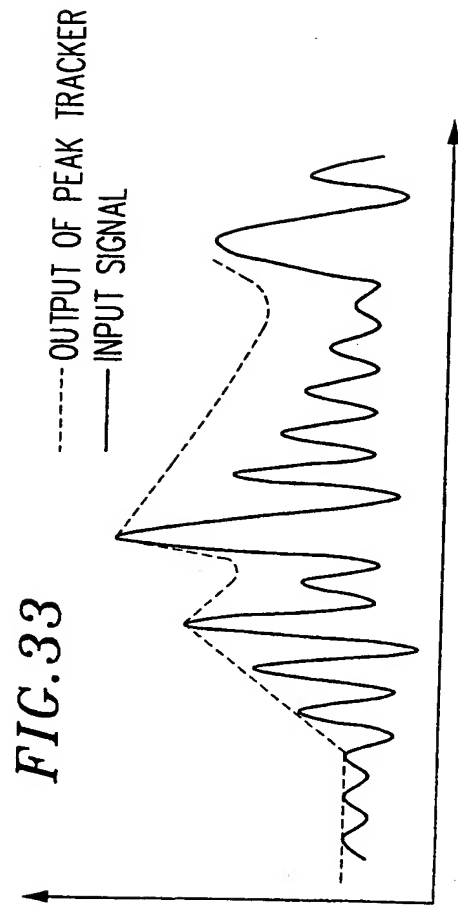
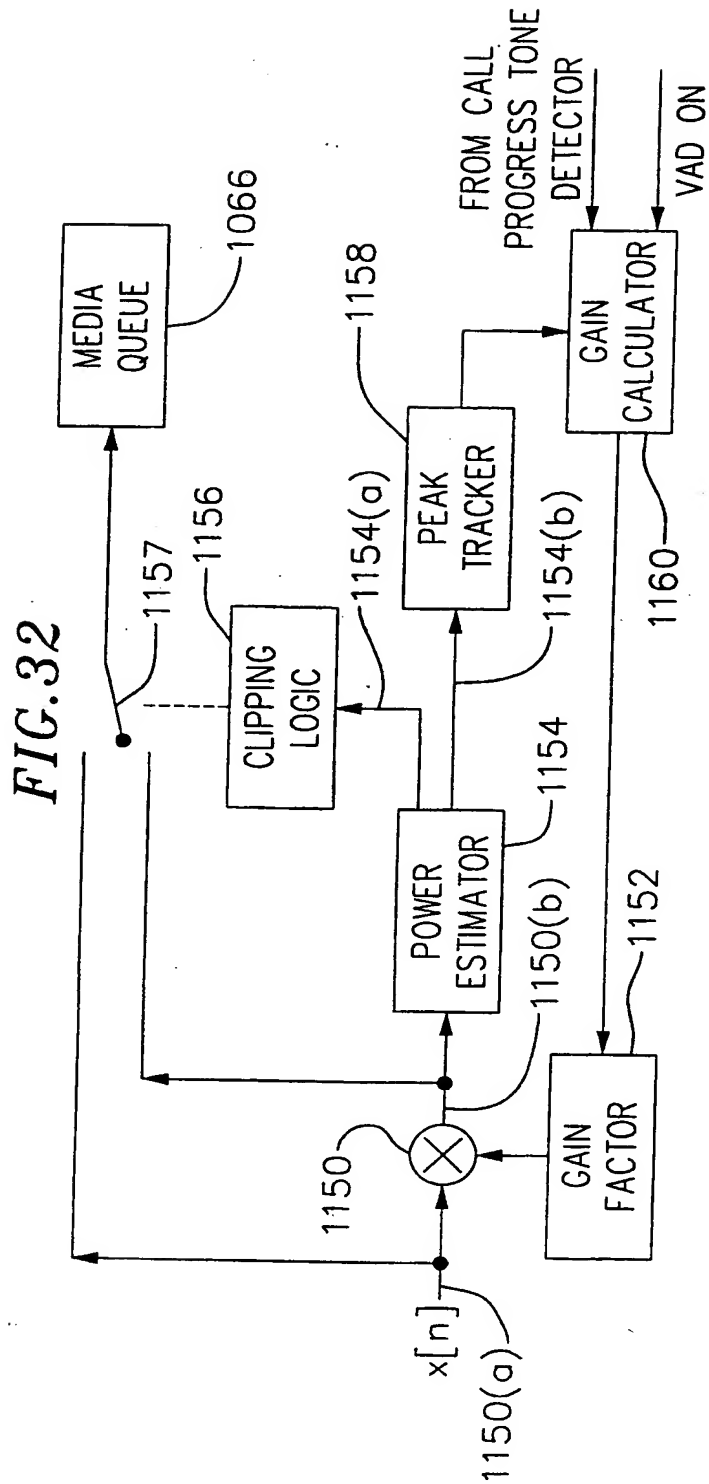
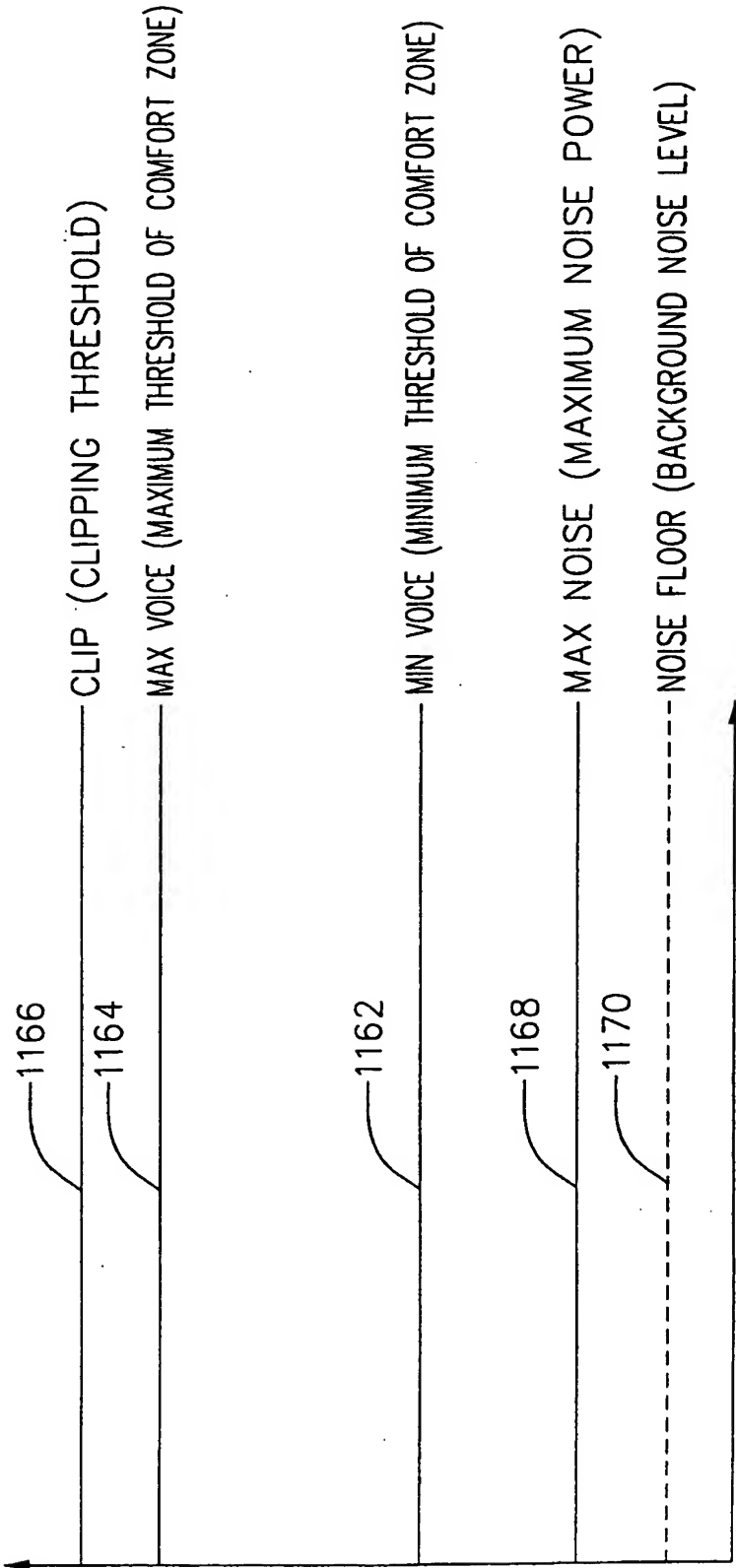
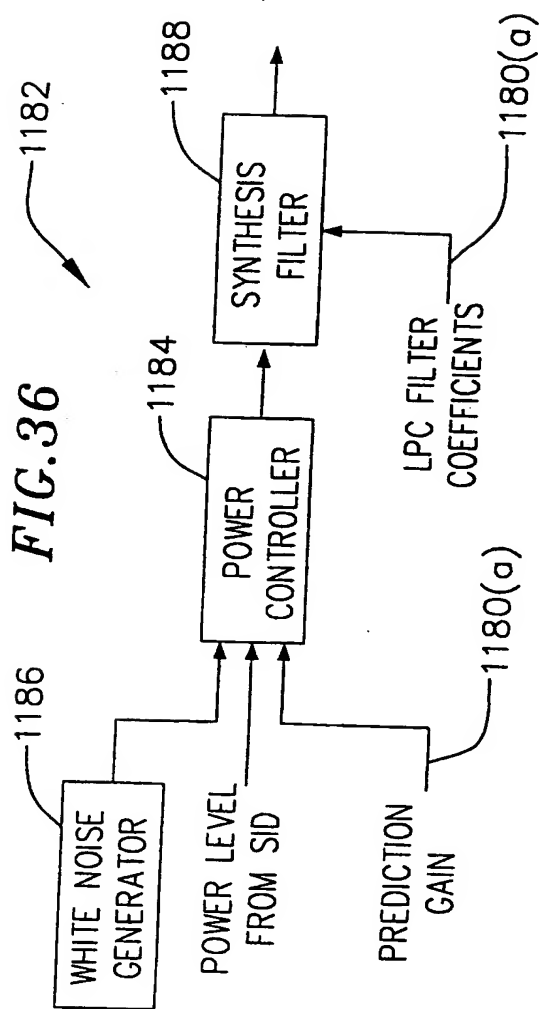
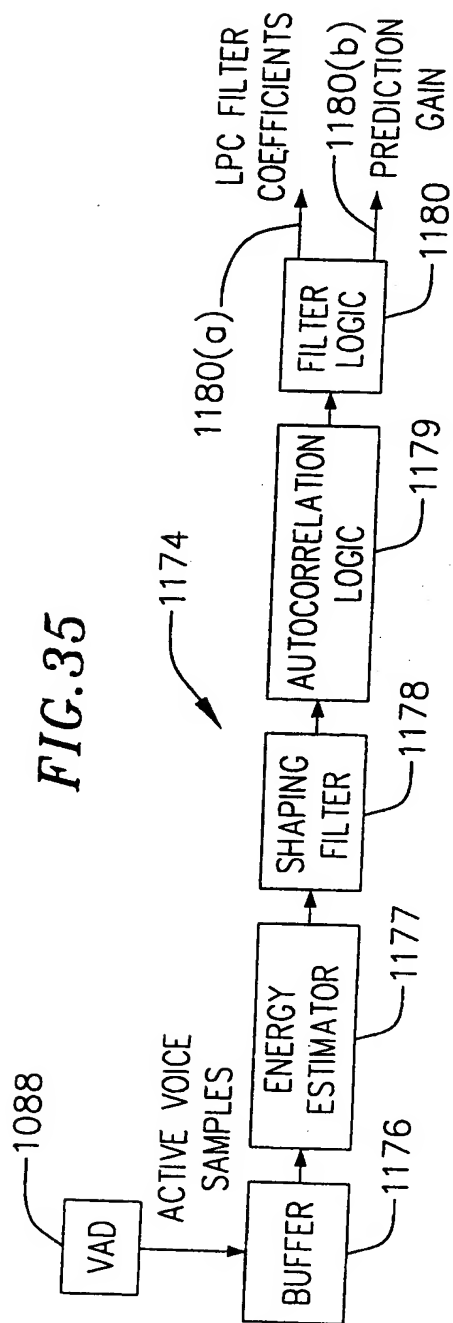


FIG.34







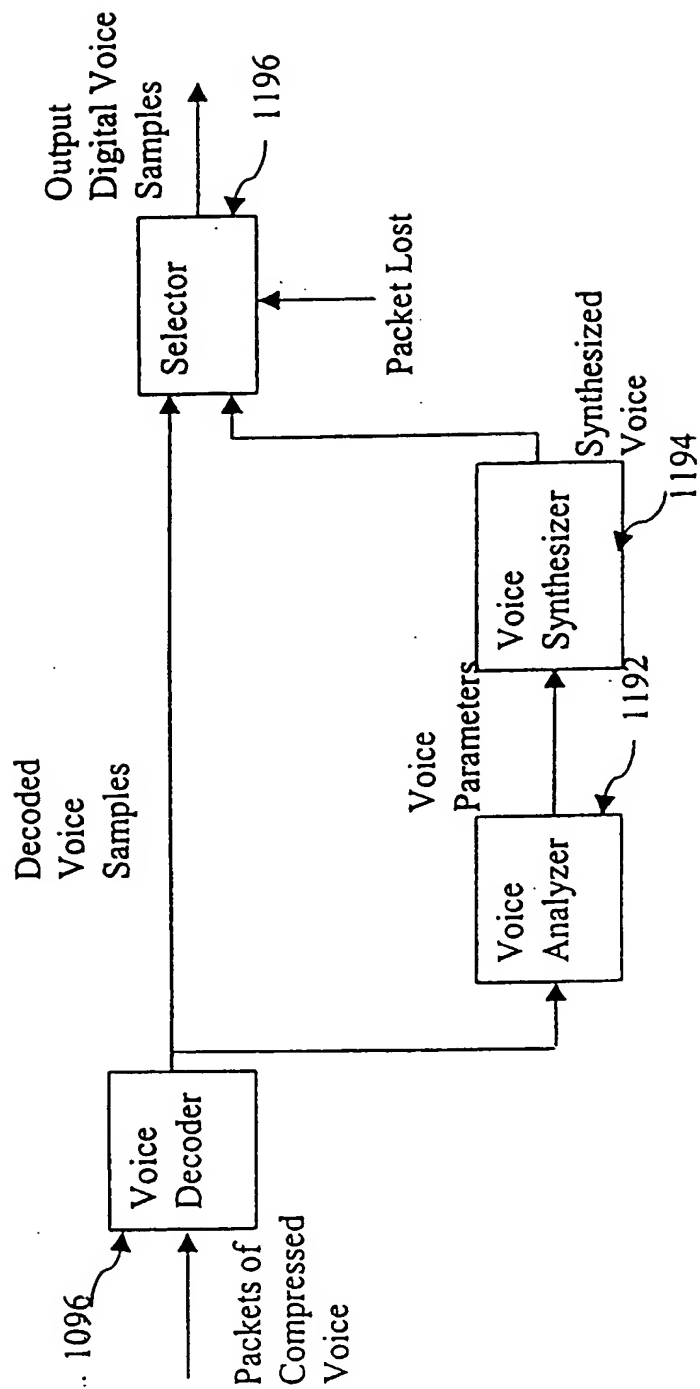


FIG. 37

FIG. 38A

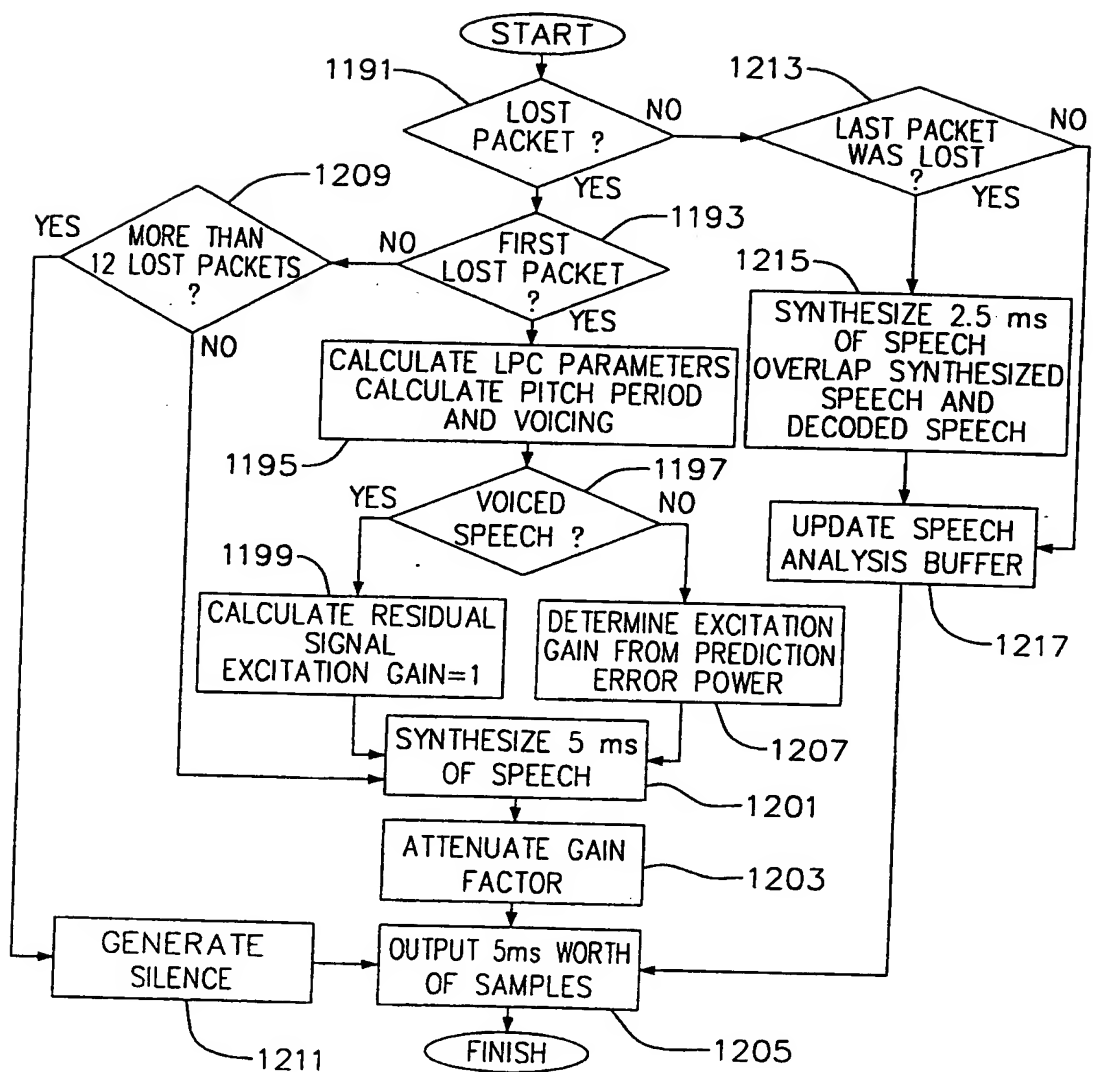


FIG. 38B

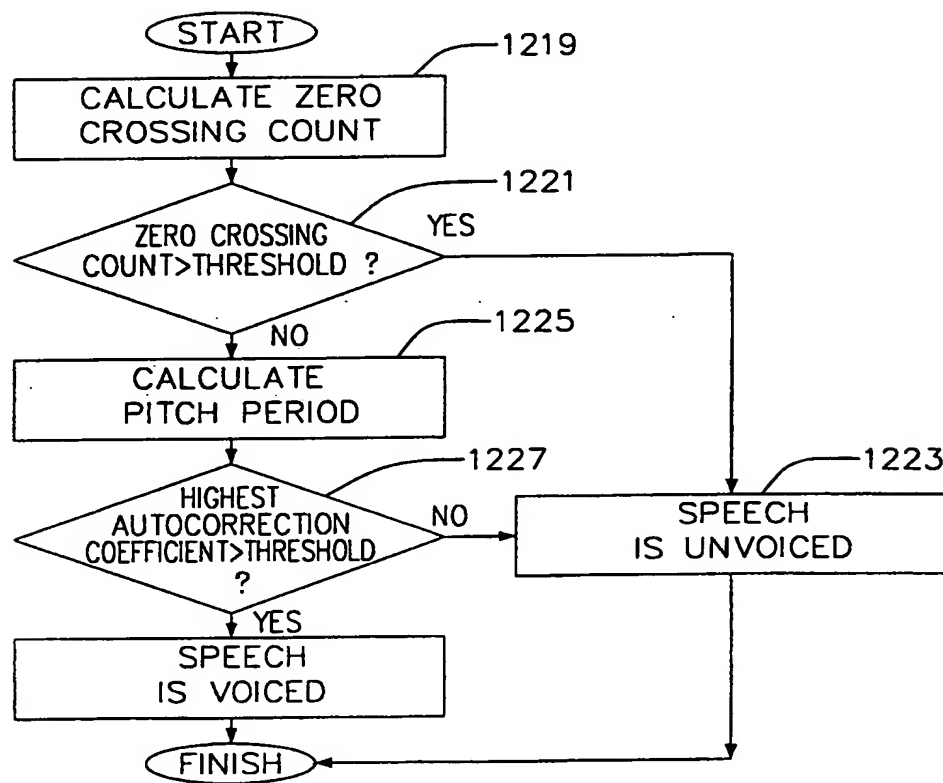


FIG. 38C

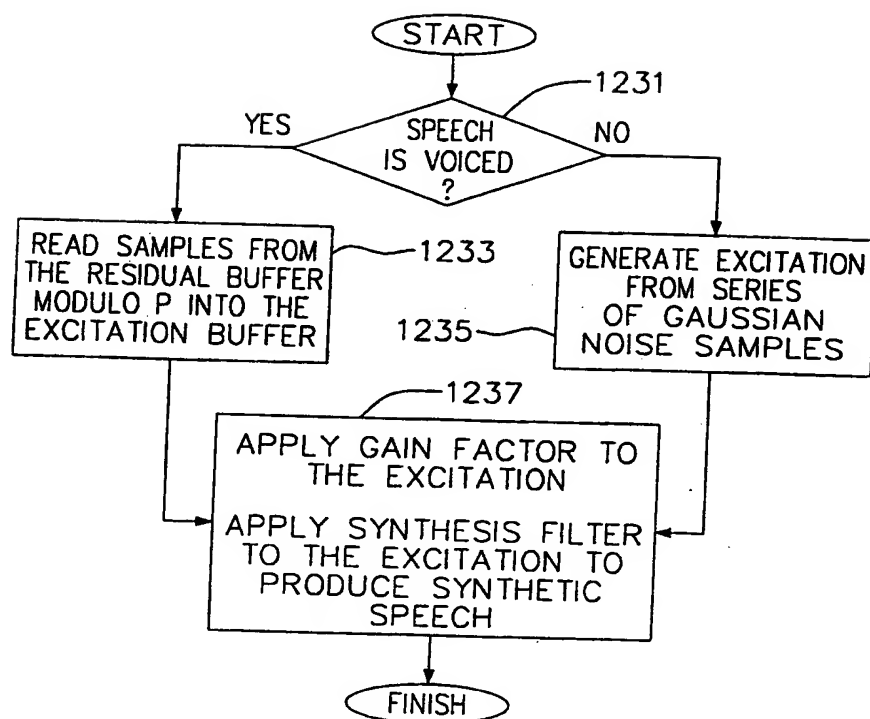
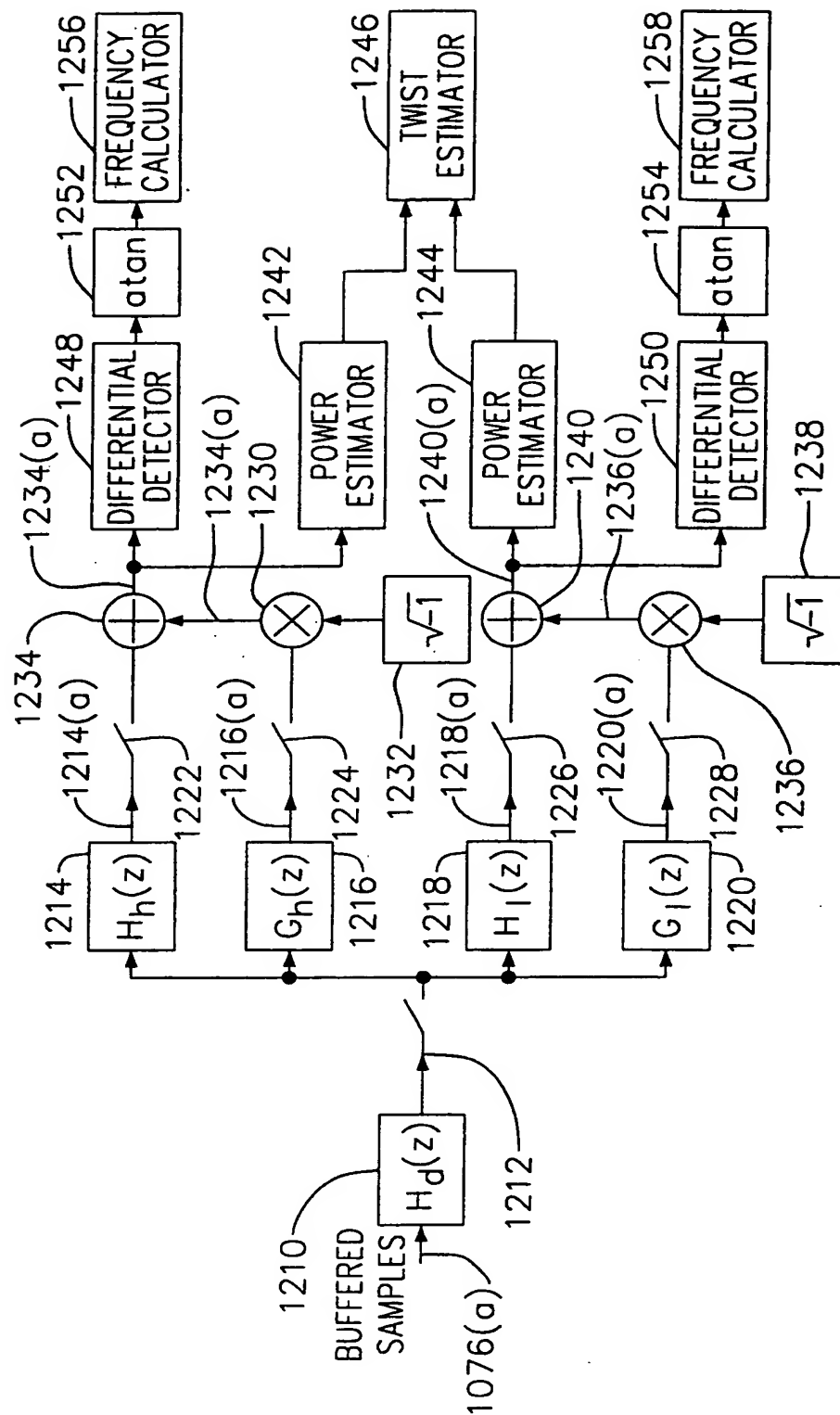


FIG. 39



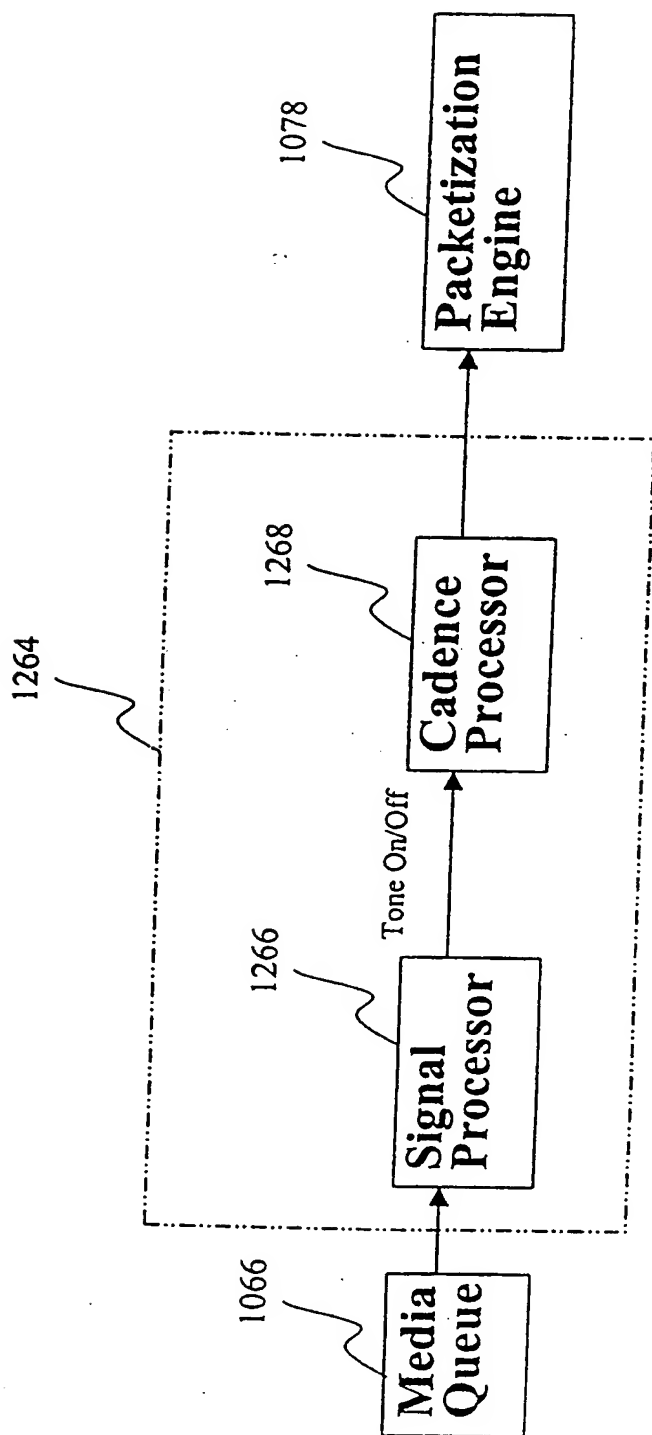
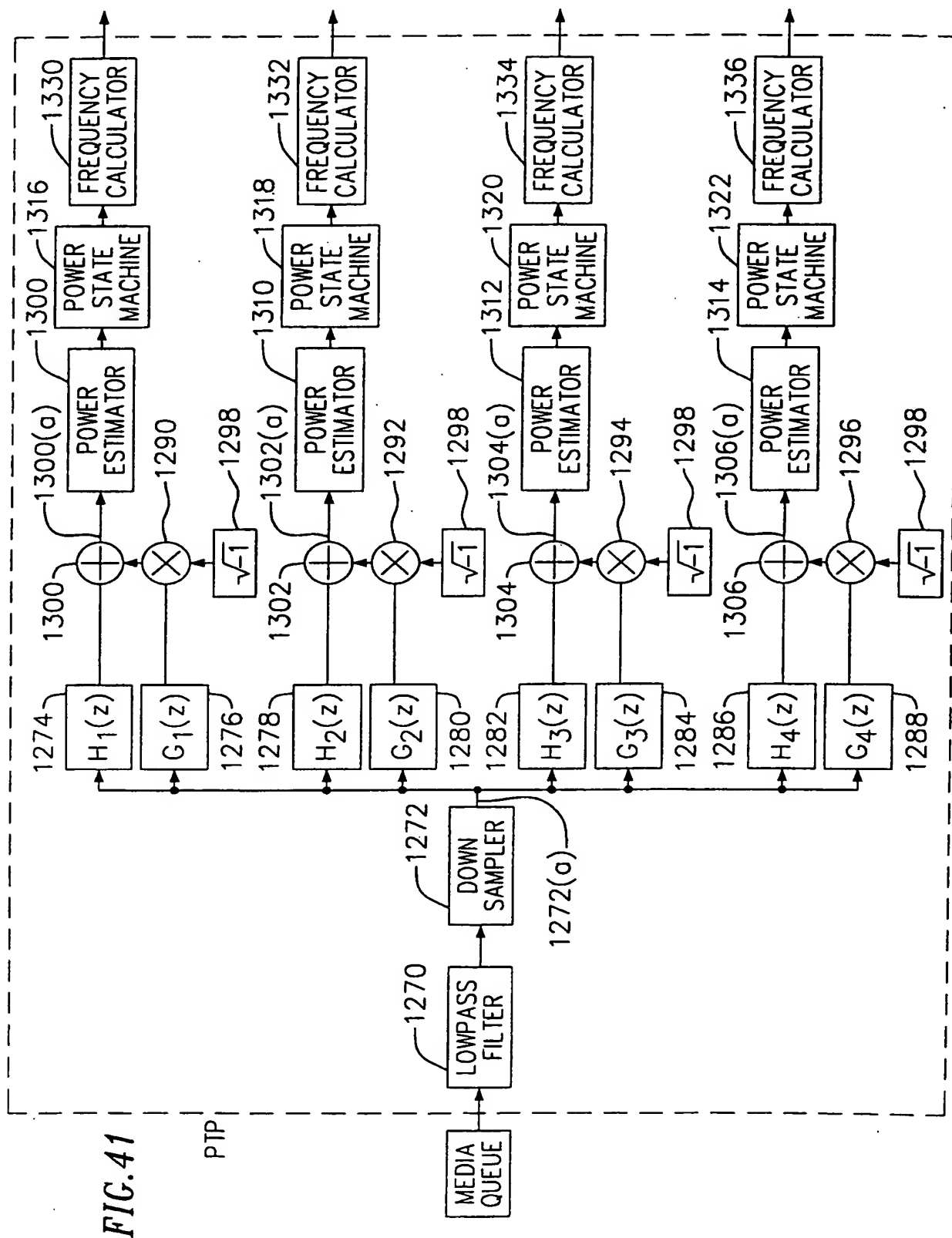


FIG. 40



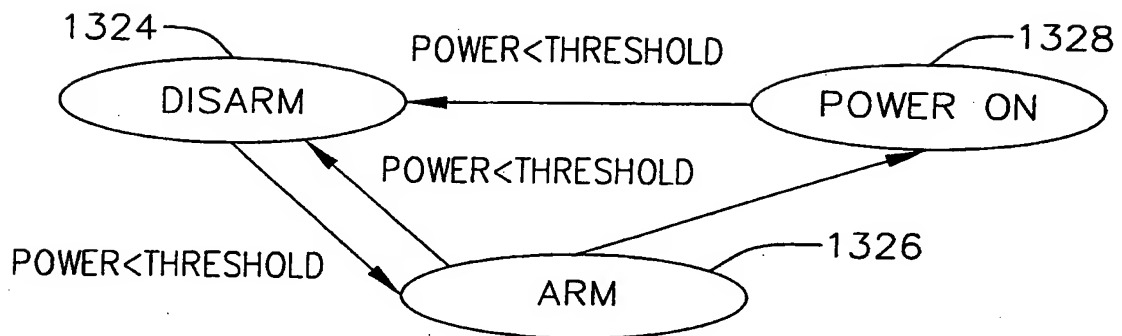
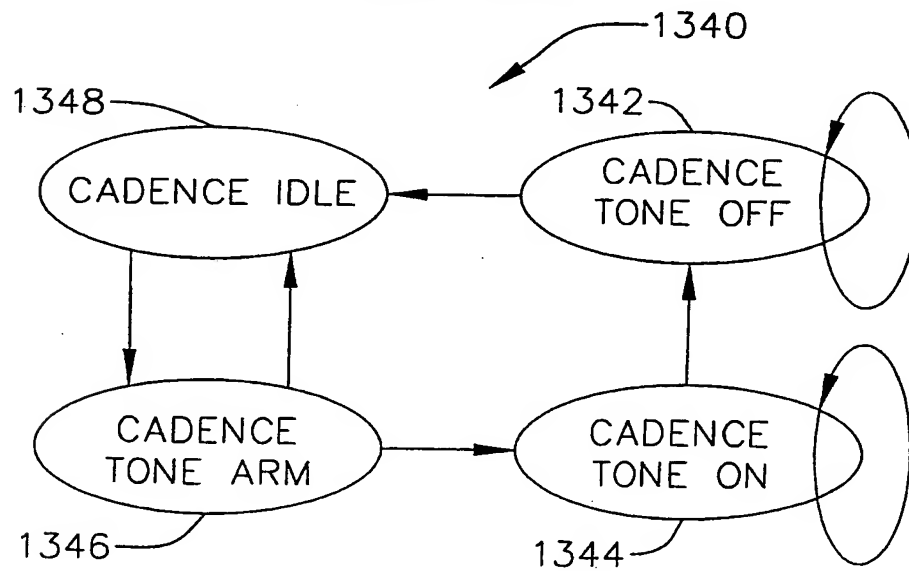
*FIG. 42**FIG. 43*



FIG. 44

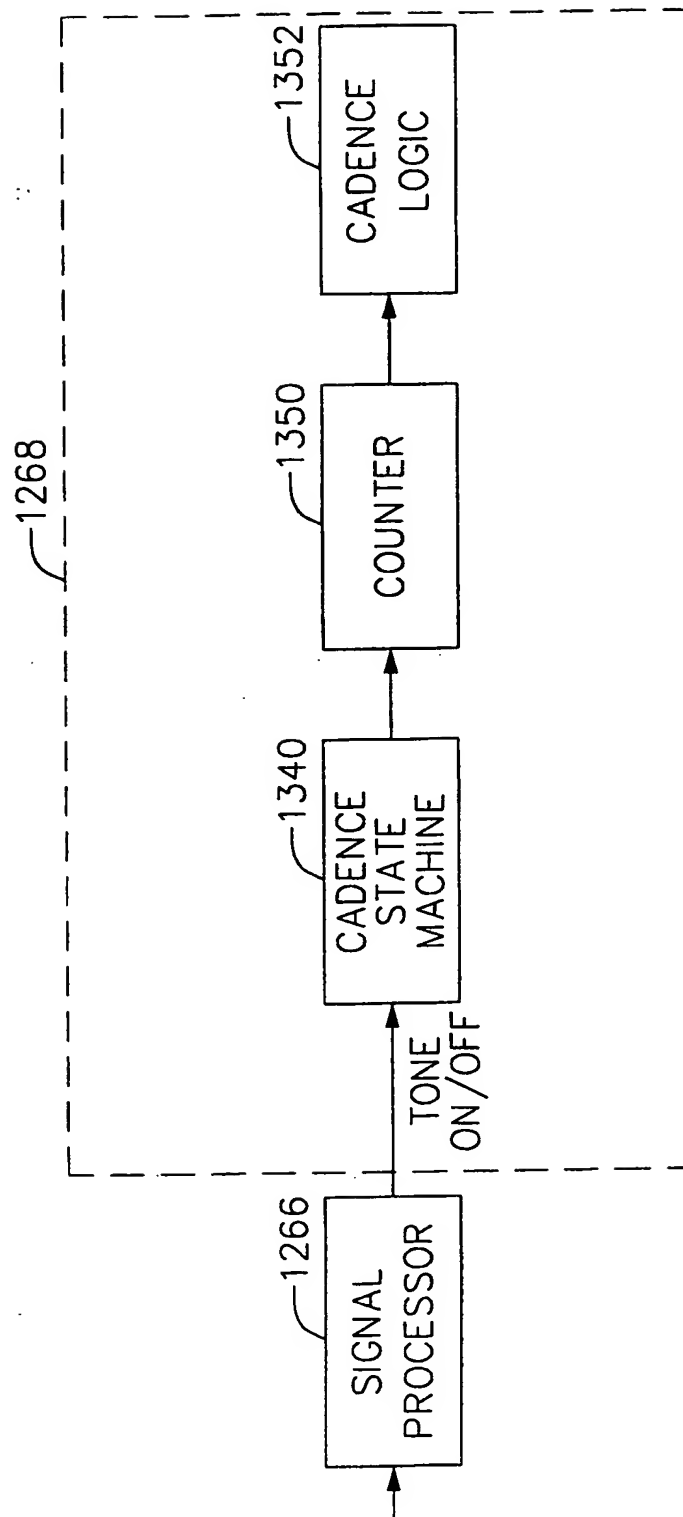


FIG. 45

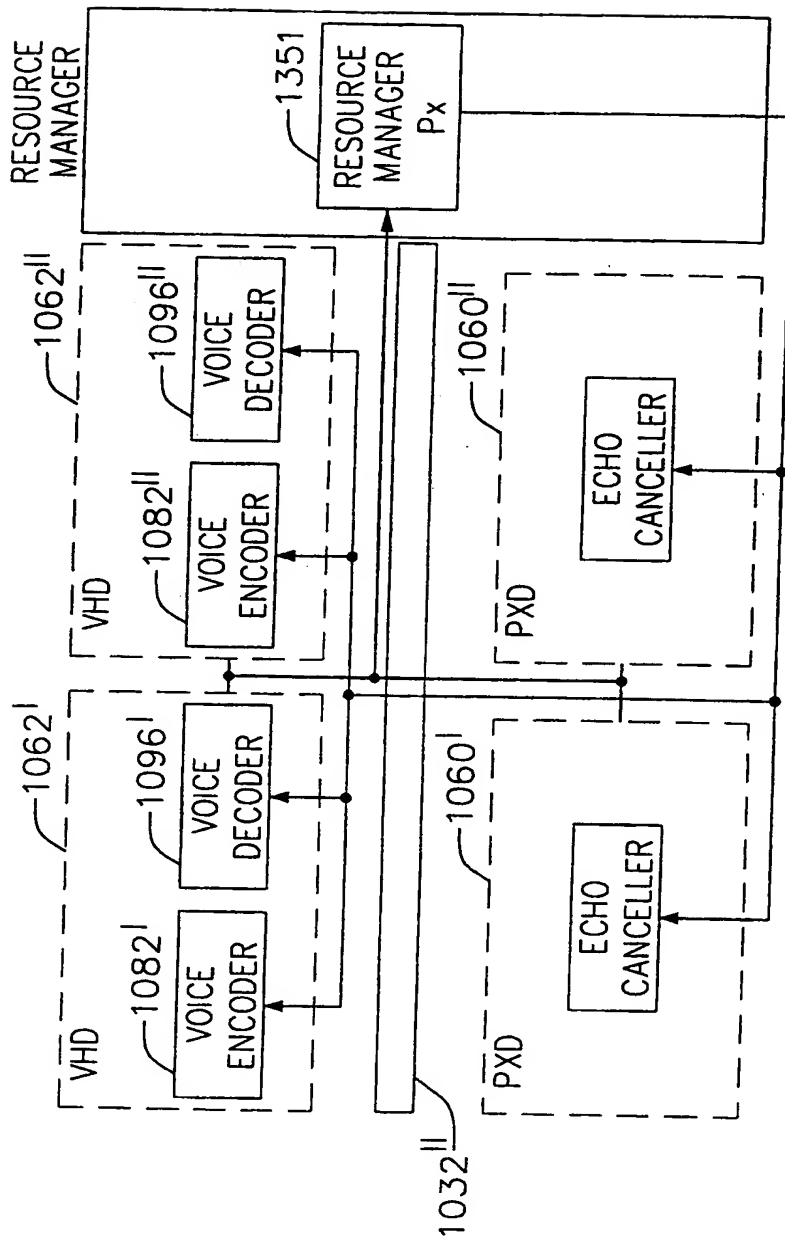


FIG. 46

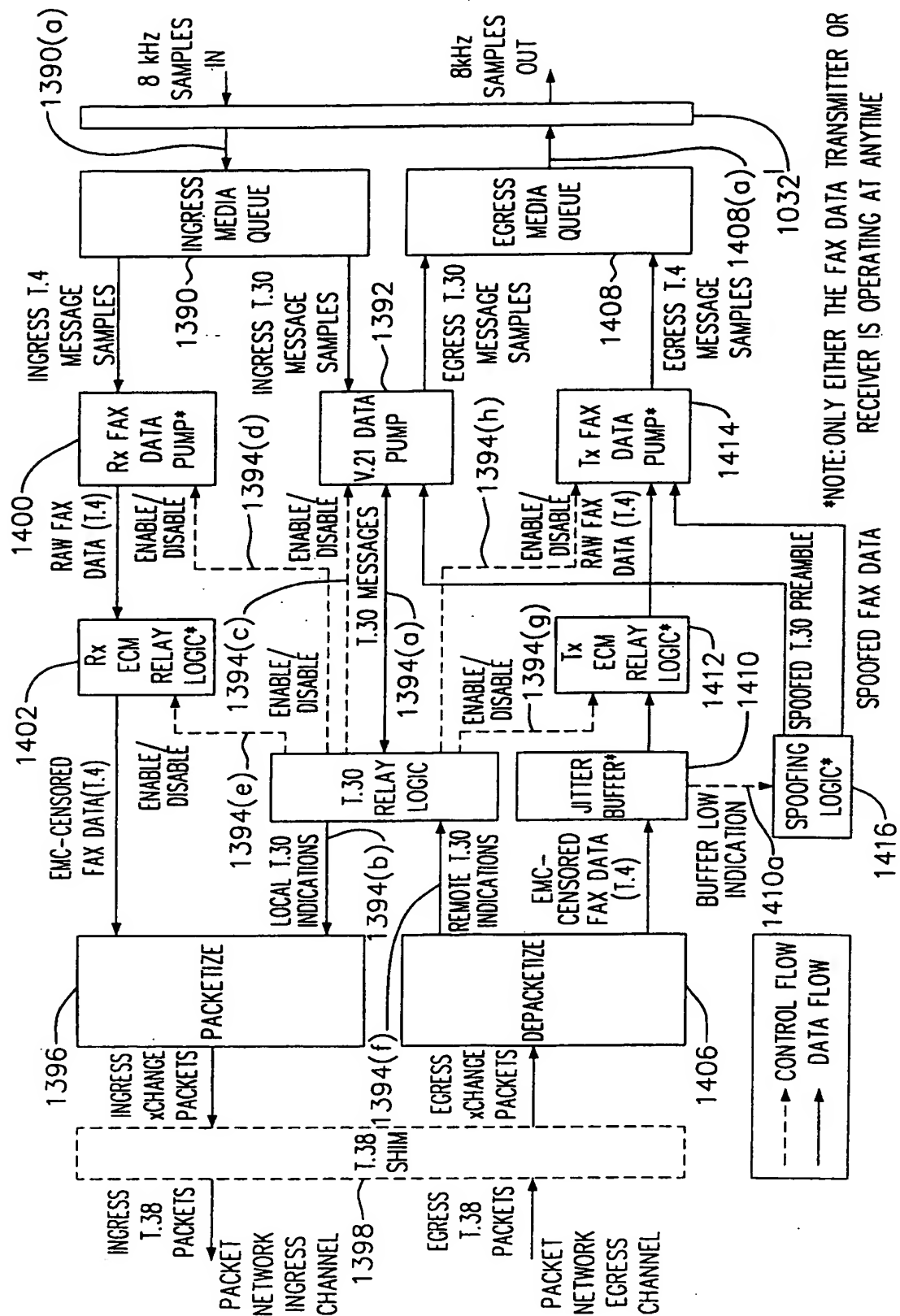
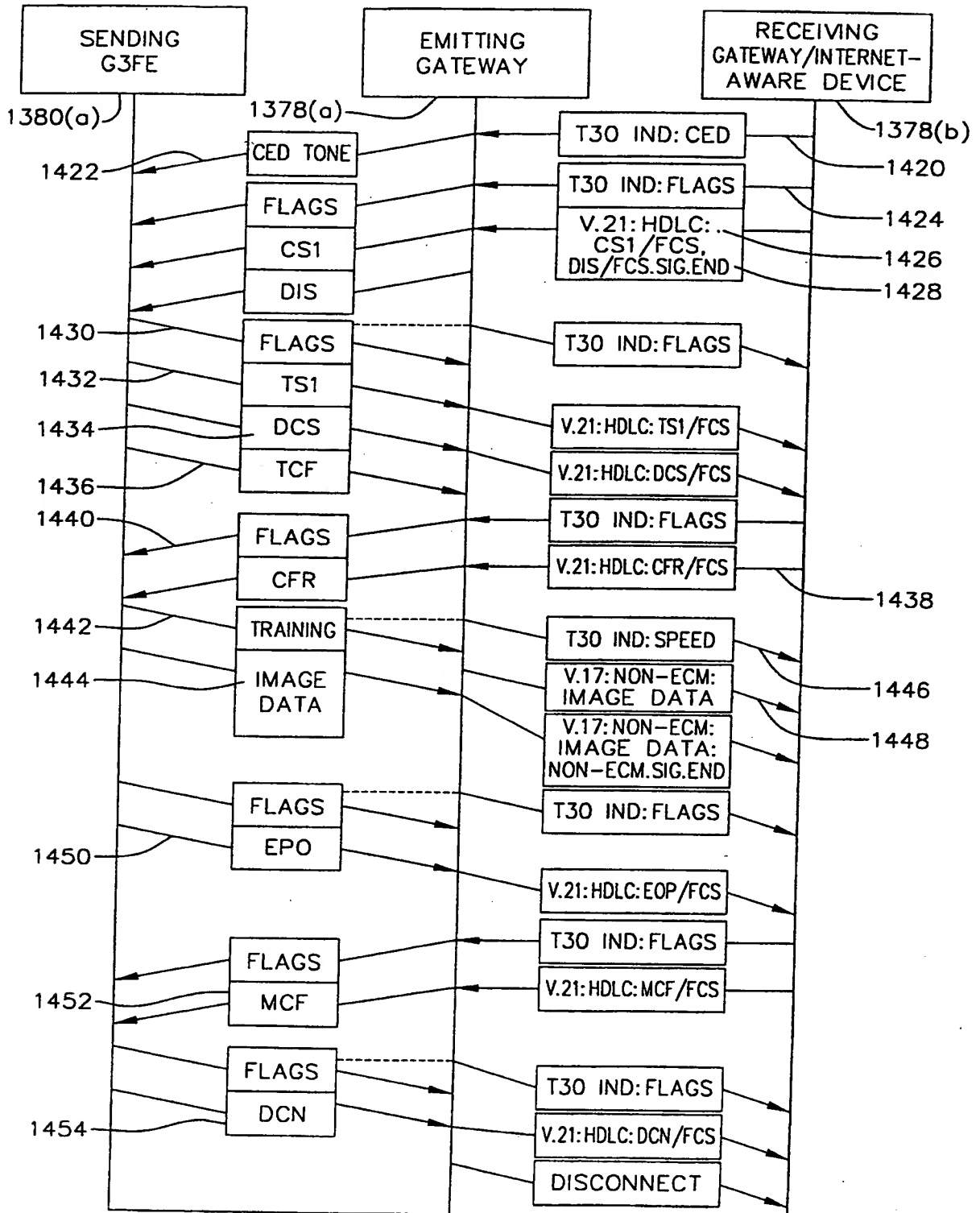


FIG. 47



SUBSTITUTE SHEET (RULE 26)

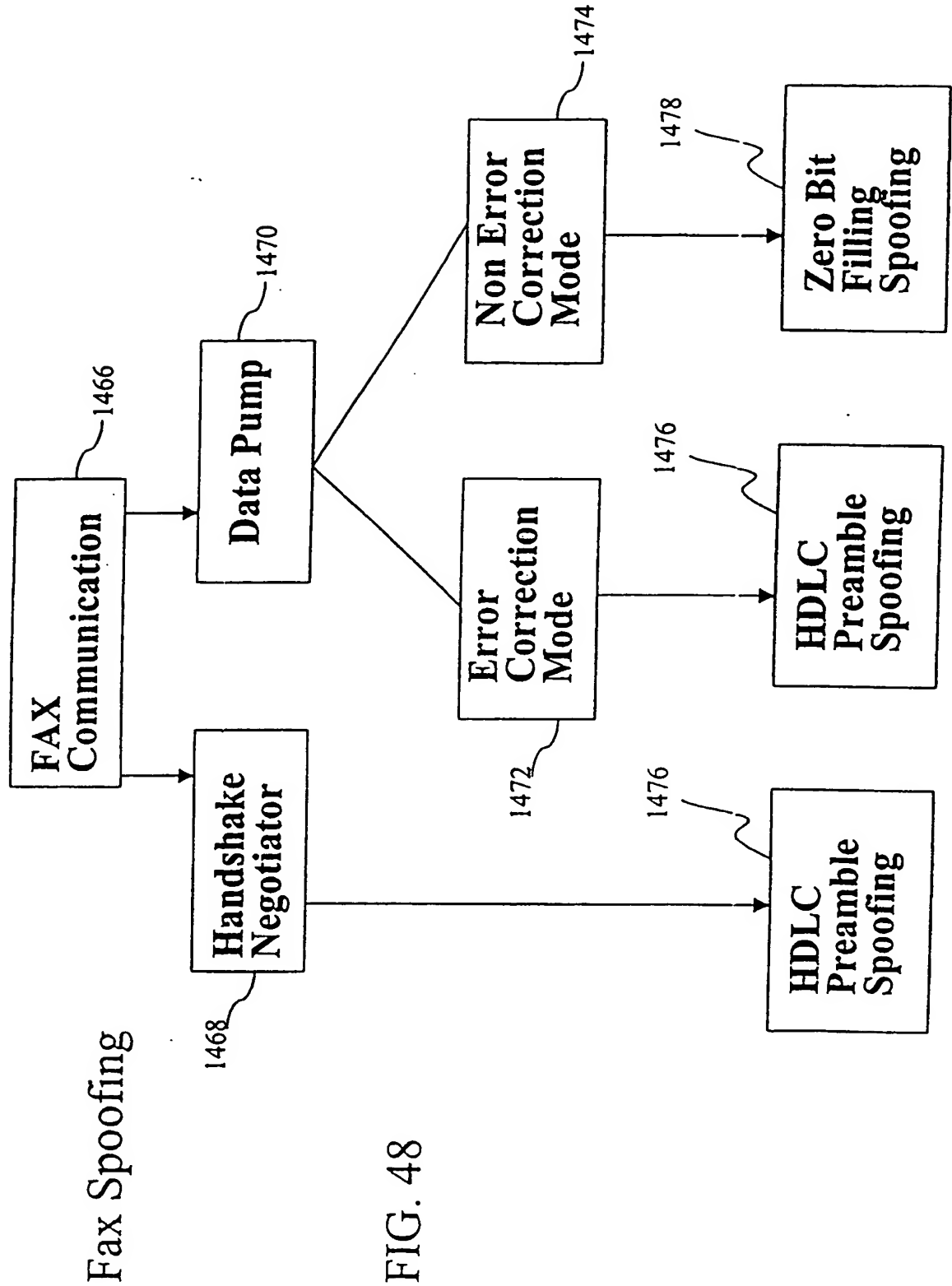


FIG. 48

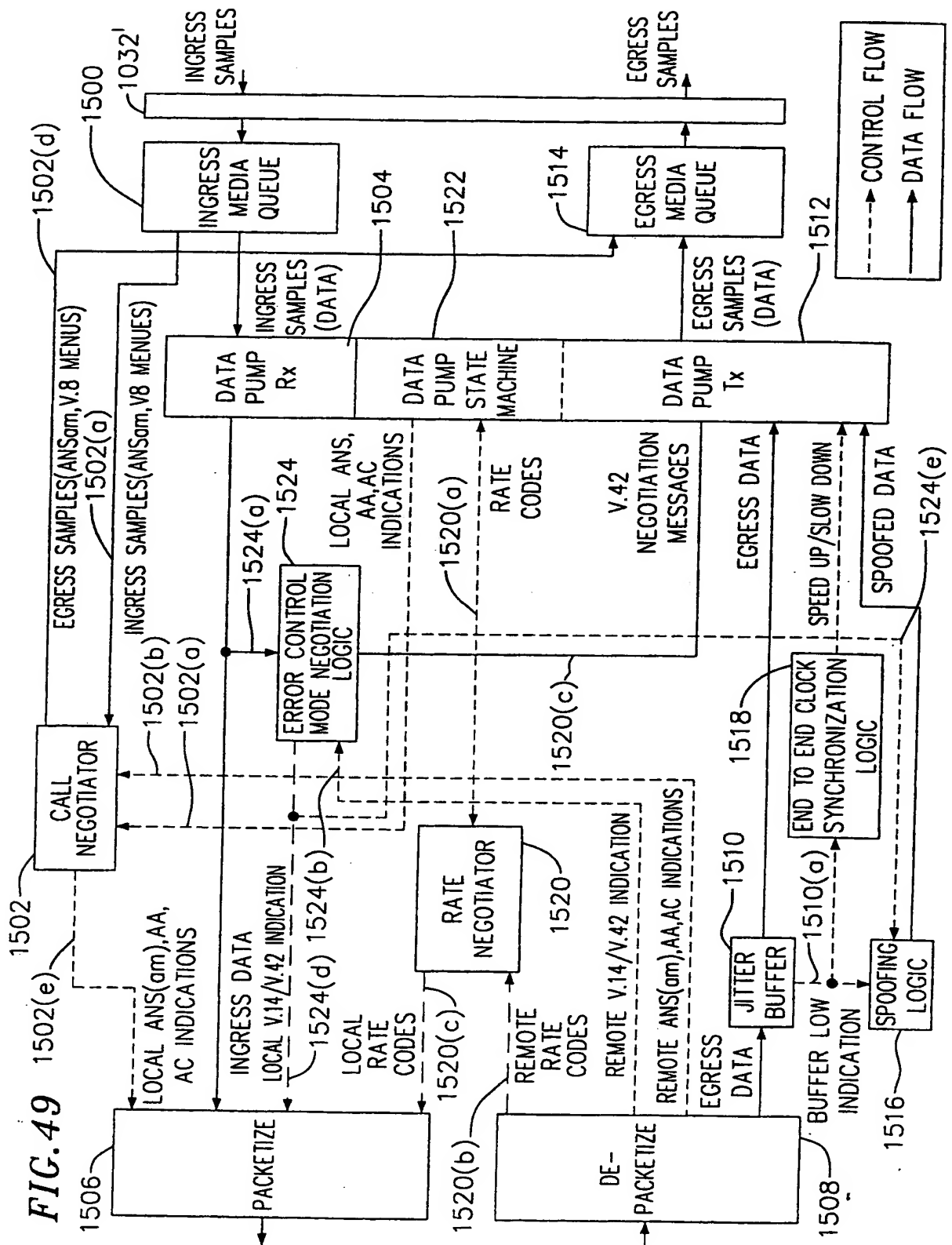
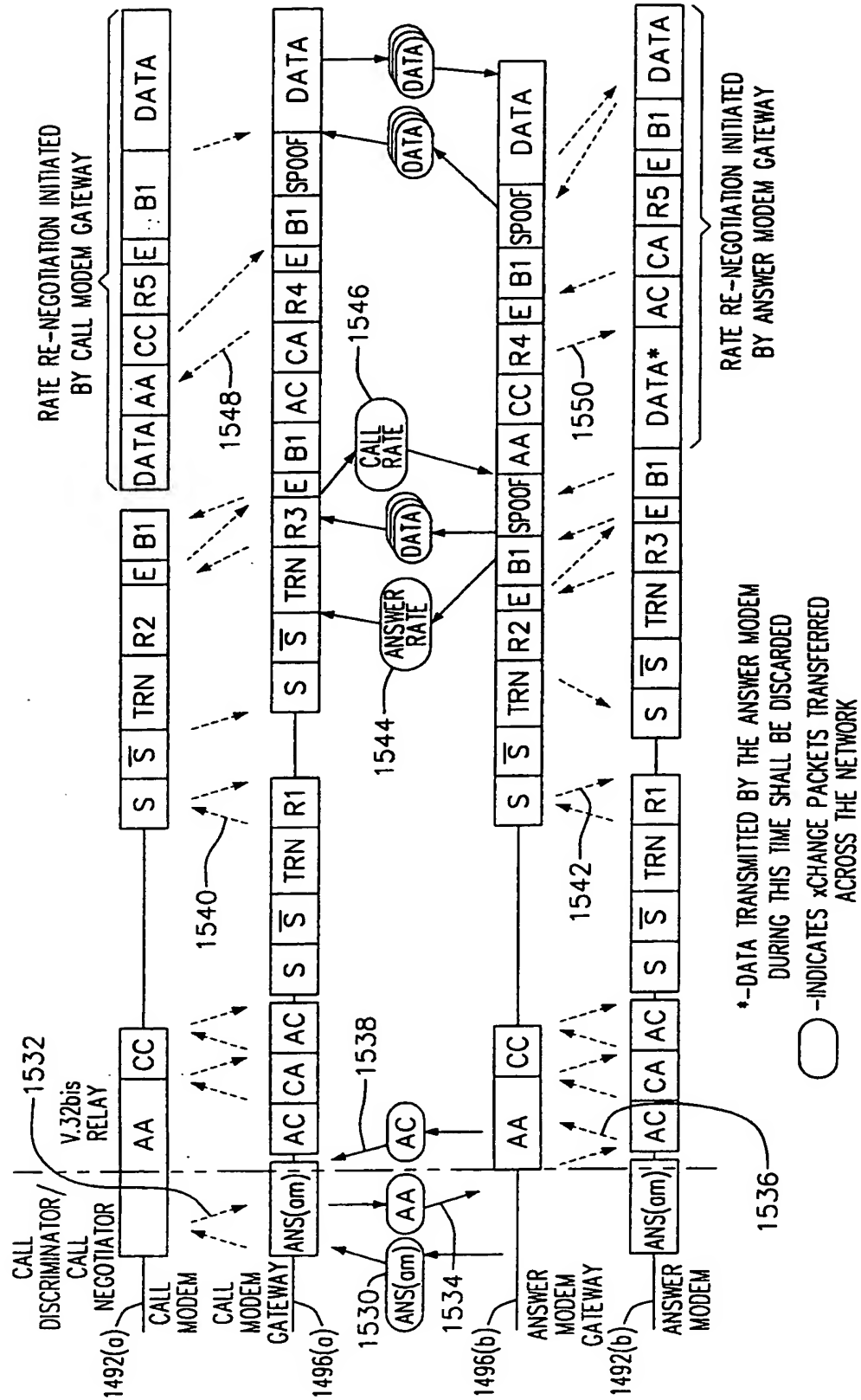


FIG. 50







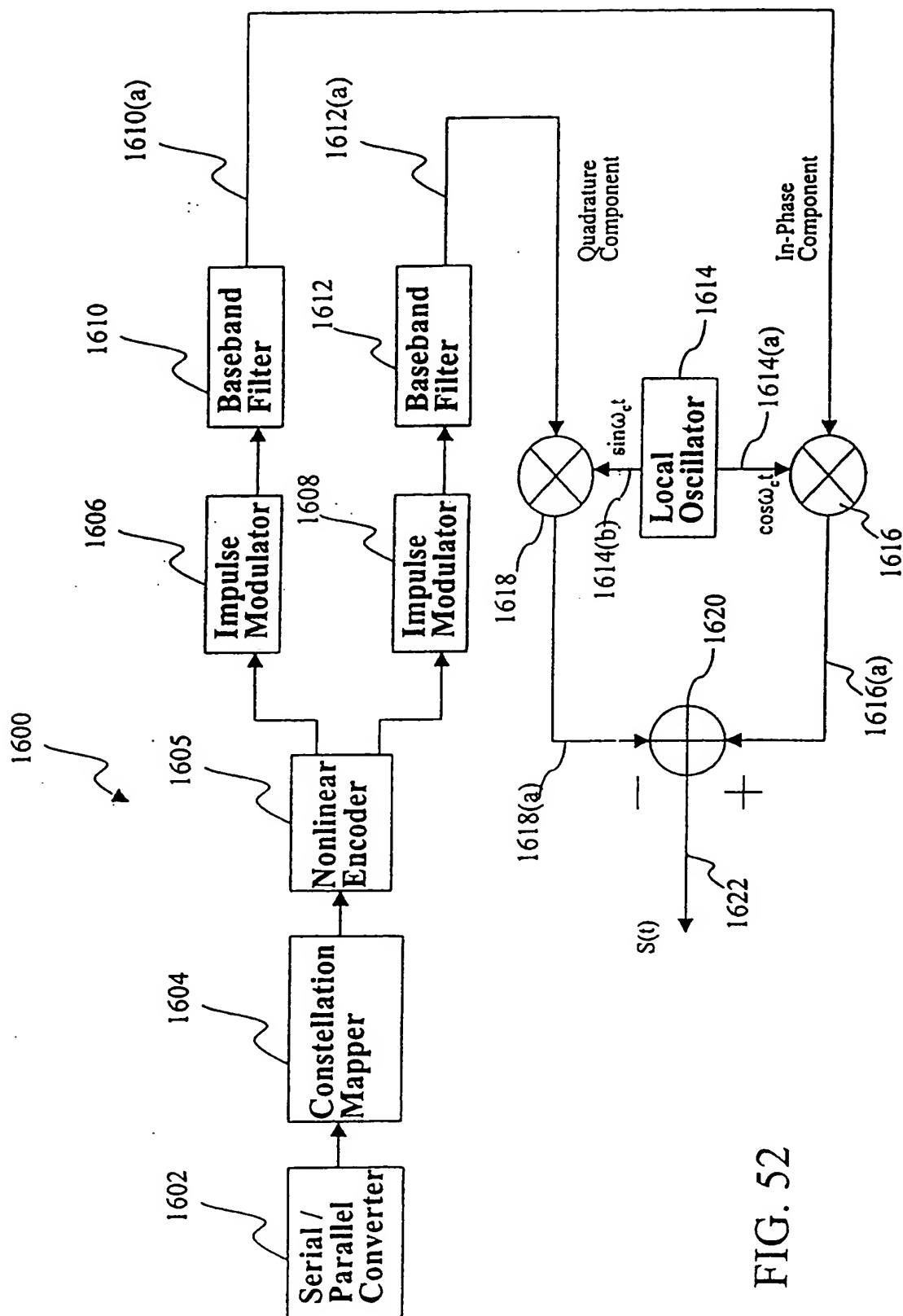


FIG. 52

FIG. 53

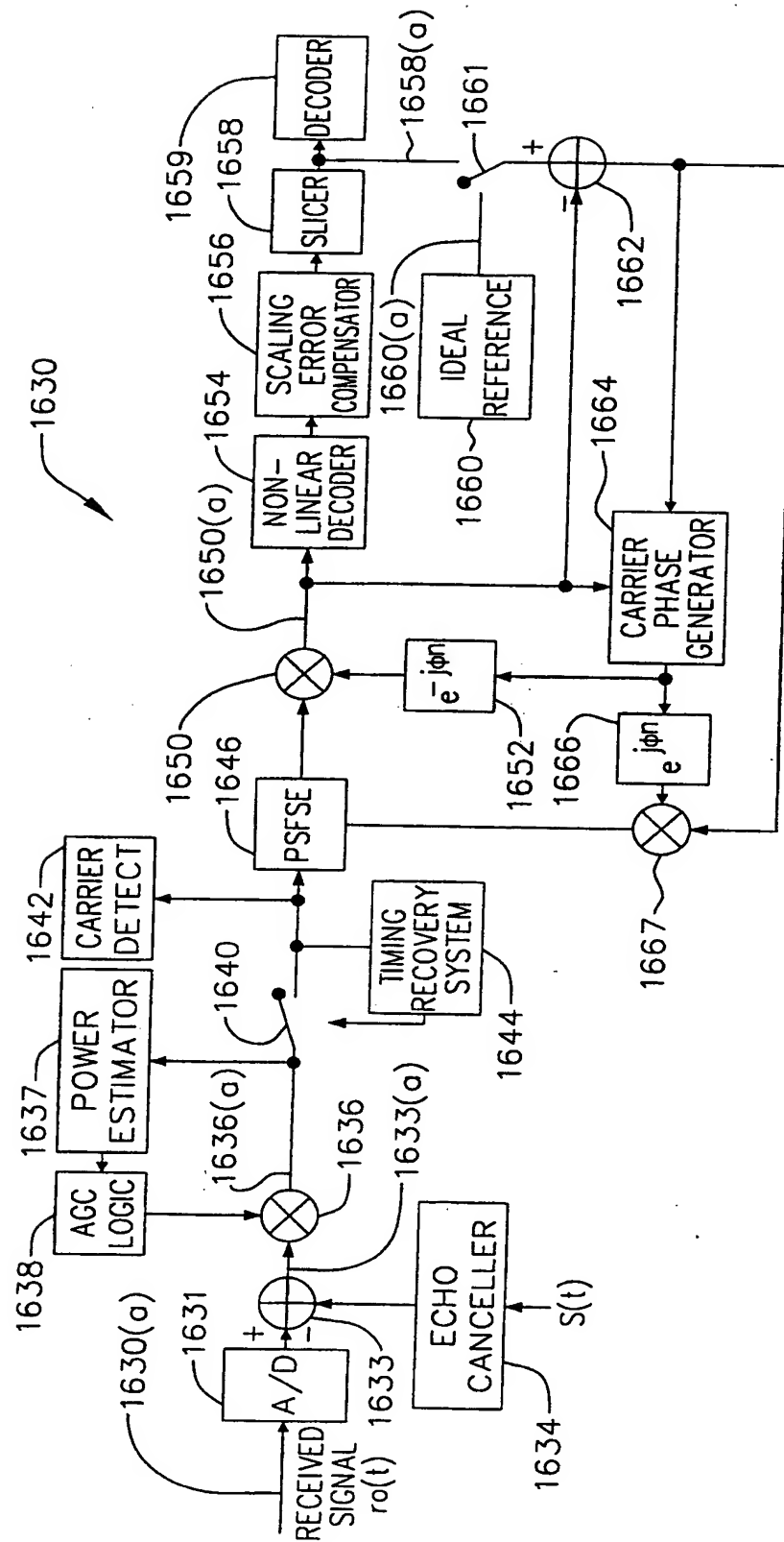


FIG. 54

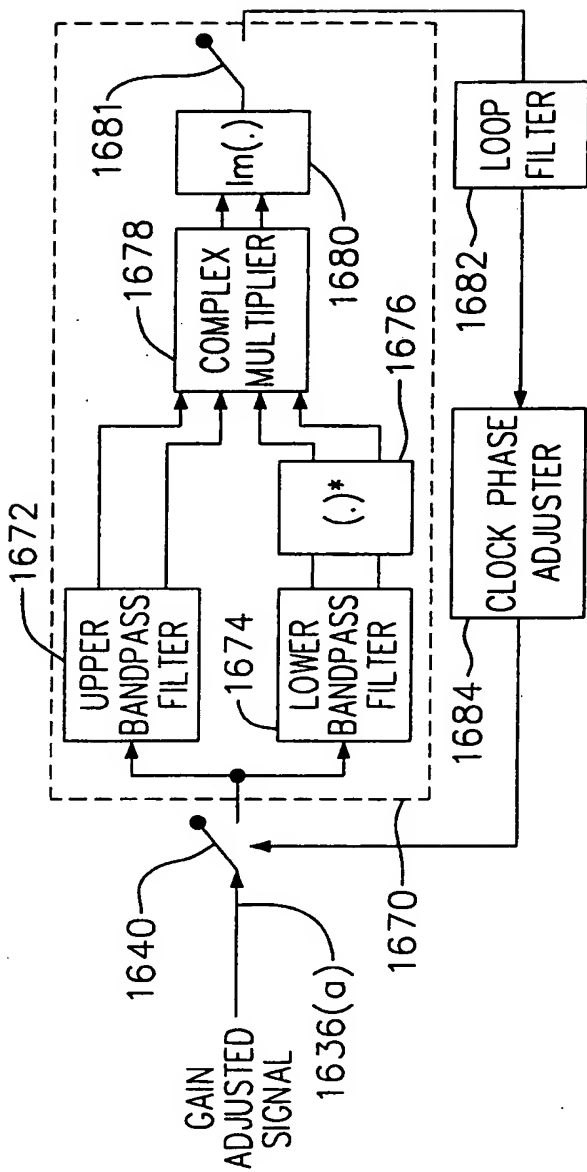
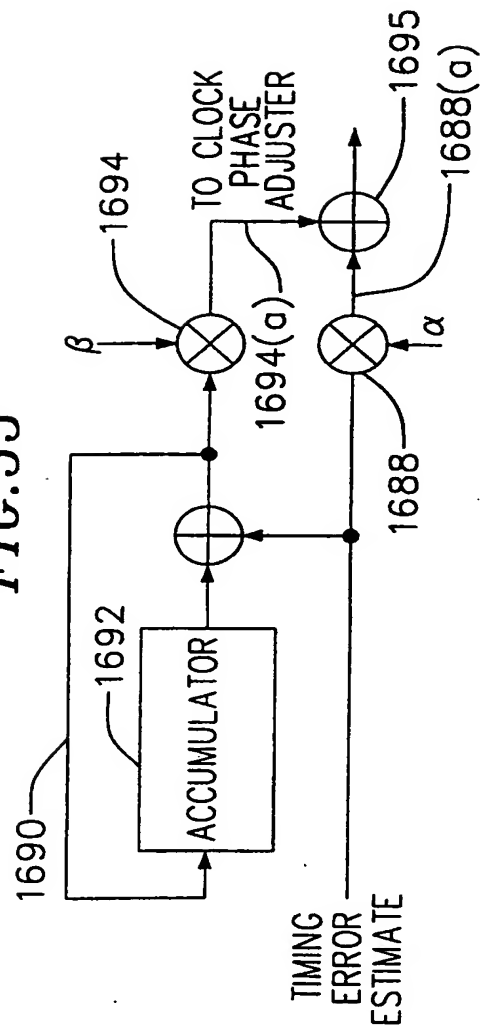
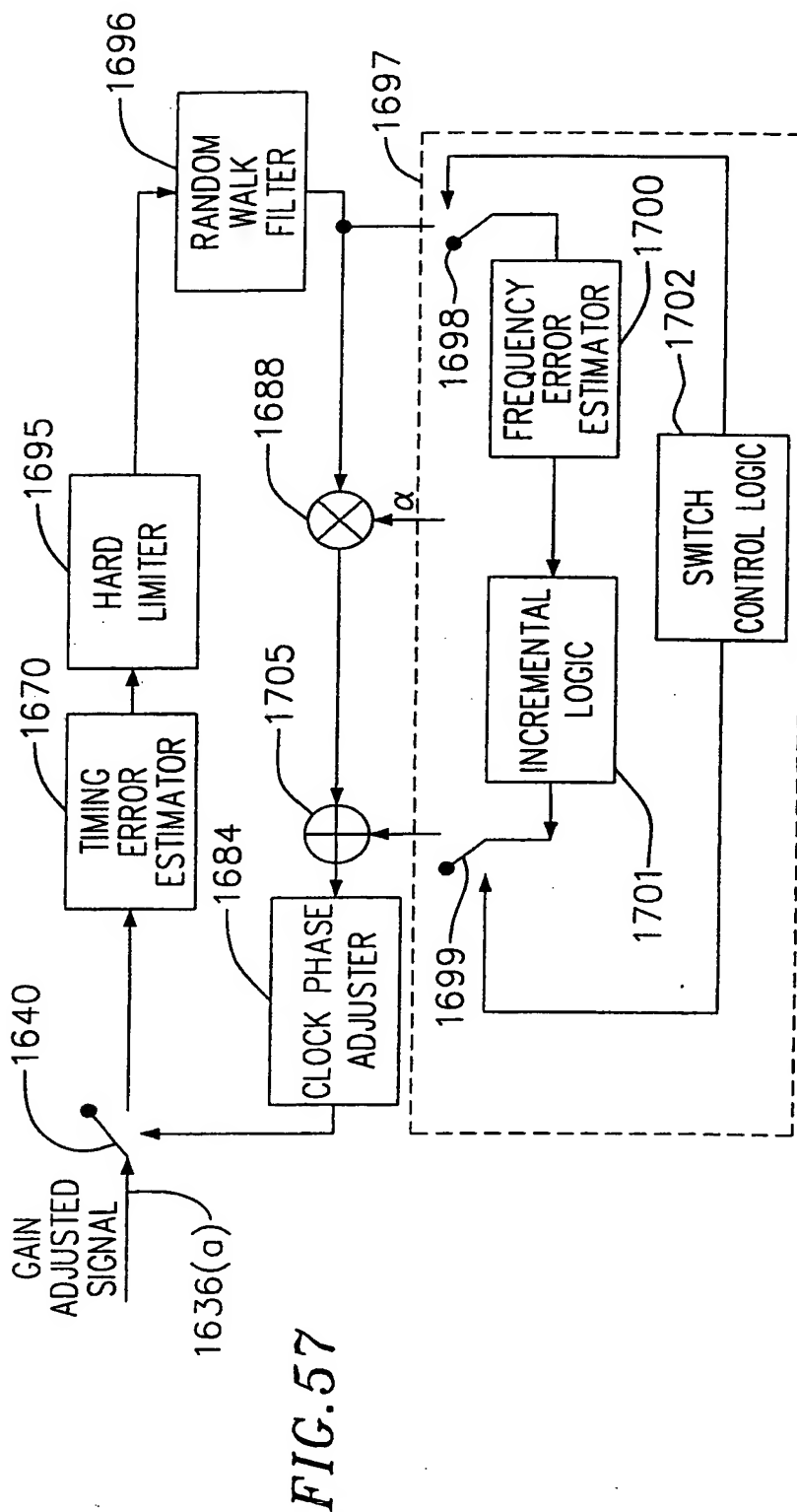
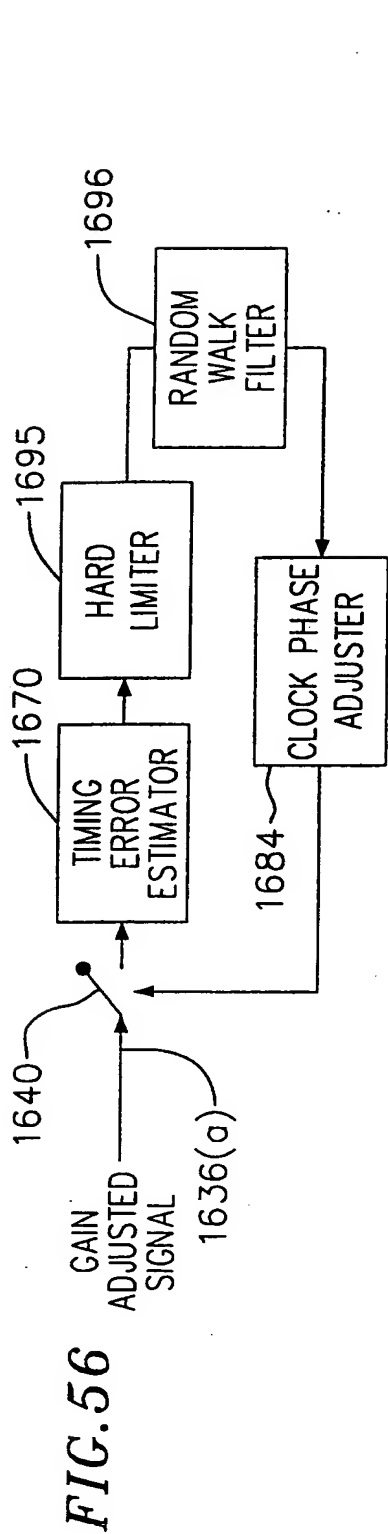


FIG. 55





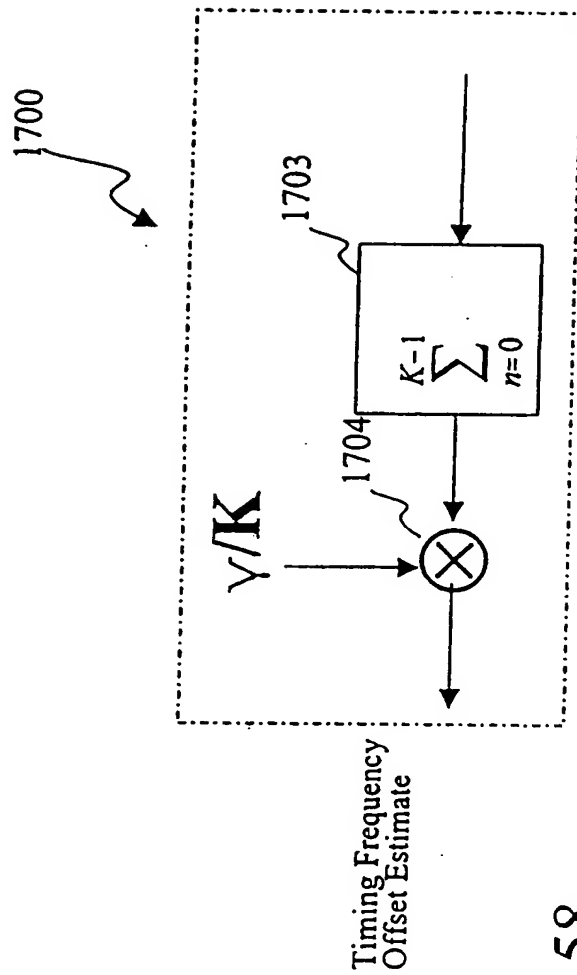


FIG. 58

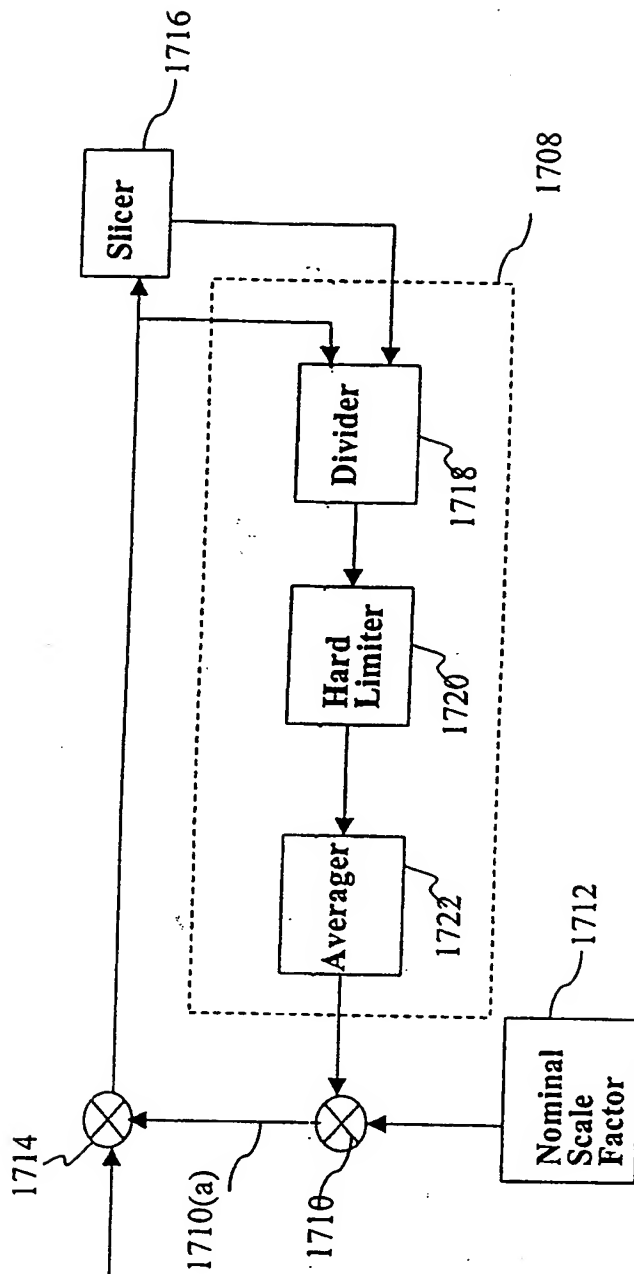


FIG. 59

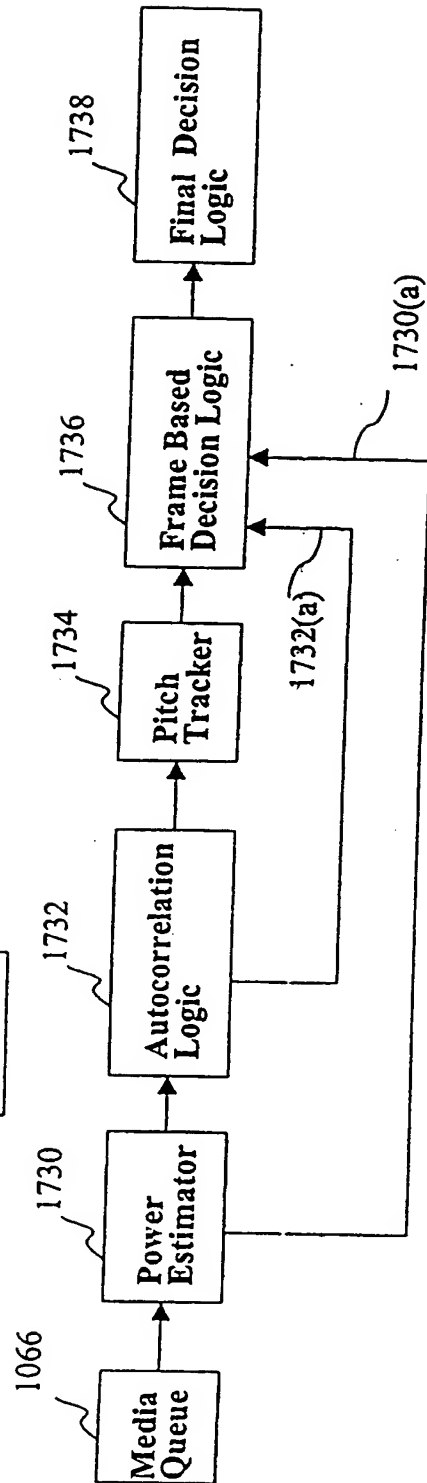


FIG. 60